ESCUELA TÉCNICA SUPERIOR DE INGENIEROS
INDUSTRIALES Y DE TELECOMUNICACIÓN

UNIVERSIDAD DE CANTABRIA

Proyecto Fin de Carrera

GESTOR DE CONTENIDOS PARA LA WEB CORPORATIVA DE LA UC OPTIMIZADO PARA ACCEDER DESDE DISPOSITIVOS MÓVILES

(Content Management Service (CMS) optimized for accessing University of Cantabria's Corporative Web-site with mobile devices)

Para acceder al Titulo de

INGENIERO TÉCNICO DE TELECOMUNICACIÓN

Autor: Pablo Pérez Capa

Octubre - 2012

INGENIERÍA TÉCNICA DE TELECOMUNICACIÓN

CALIFICACIÓN DEL PROYECTO FIN DE CARRERA

Realizado por: Pablo Pérez Capa
Director del PFC: José María Zamanillo Sainz de la Maza

Título: "Gestor de contenidos para la Web Corporativa de la UC optimizado para acceder desde dispositivos móviles"

Title: "Content Management Service (CMS) optimized for accessing University of Cantabria's Corporative Web-site with mobile devices"

Presentado a examen el día: 29 de Octubre de 2012

para acceder al Título de

INGENIERO TÉCNICO DE TELECOMUNICACIÓN, ESPECIALIDAD EN SISTEMAS ELECTRÓNICOS

Composición del Tribunal:
Presidente: Fernández Ibáñez, Tomás
Secretario: Zamanillo Sainz de la Maza, José María
Vocal: Fanjul Vélez, Félix

Este Tribunal ha resuelto otorgar la calificación de: .....

Fdo.: El Presidente	Fdo.: El Secretario
Fdo.: El Vocal	Fdo.: El Director del PFC (sólo si es distinto del Secretario)
Vº Bº del Subdirector	Proyecto Fin de Carrera Nº (a asignar por Secretaría)

AGRADECIMIENTOS

A Francisco Ramos y a Jose María Zamanillo por su ayuda e implicación a la hora de realizar este proyecto.

A mi familia, por su incondicional apoyo durante estos años. Se os quiere pese a todo, pero habrá que buscar nuevos temas de conversación.

Y muy especialmente a María, por todo.

ÍNDICE

Tabla de contenido

Listado de Figuras

1. RESUMEN

1.1. RESUMEN

Actualmente, el uso de dispositivos móviles está cada vez más extendido entre todo tipo de usuarios, por lo que hacer buen uso de esta nueva y creciente vía de comunicación es fundamental para transmitir contenidos de forma eficaz. De este modo, y ante la carencia actual de cualquier tipo de soporte para este tipo de plataformas, se plantea el desarrollo de un gestor para los contenidos de la web corporativa de la Universidad de Cantabria que permita su correcta visualización y fácil acceso desde estos dispositivos. Para ello, se empleará un diseño basado en arquitectura de tres capas, consistente en una base de datos (SQL Server), la lógica de negocio necesaria para acceder a los datos (programada en C#) y, por último, la capa de presentación, que en este caso consistirá en una implementación específica para dispositivos iOS (iPhone e iPad) en lenguaje Objective-C. Este tipo de arquitectura permitirá en un futuro poder hacer implementaciones para otros entornos (Android, Windows Phone, Symbian...) teniendo simplemente que crear la última de las tres capas, ya que las dos primeras tendrán total independencia del dispositivo y podrían utilizarse sin necesidad de hacer modificaciones de ningún tipo.

Palabras clave: Sistema de gestión de contenido, Arquitectura de 3-capas, Servicios Web, iOS.

1.2. ABSTRACT

Nowadays, the use of mobile devices is increasingly spreading among all users, so making good use of this new and growing way of communication is essential to effectively share contents. So, given the current lack of any support for these platforms, comes the idea of developing a content manager (CMS) for the corporate website of the University of Cantabria to allow proper display and easy access from these devices. To achieve this, a design based on a three-tier architecture will be used, consisting of a database (SQL Server), business logic required to access data (in C#), and finally the presentation layer, which in this case will consist of a specific implementation for iOS devices (iPhone and iPad) developed in Objective-C language. In the future, this type of architecture will allow to easily do implementations for other environments (Android, Windows Phone, Symbian...) only by creating the last of the three layers, as the first two have complete independence of the device and could be used with no need for modifications of any kind.

Keywords: CMS, 3-layer Architecture, Web Services, iOS.

2. INTRODUCCIÓN

2.1. SISTEMAS DE GESTIÓN DE CONTENIDOS (CMS)

Un sistema de gestión de contenidos o CMS (*Content Management System*) es un software alojado en un servidor Web que permite controlar fácilmente el contenido de una o varias bases de datos. Se utiliza fundamentalmente para la creación, administración y publicación de información de cualquier tipo (textos, imágenes, vídeos, etc.).

Una de las características principales de los sistemas gestores de contenido es la capacidad de separar el diseño de la capa de datos en el tratamiento de un sitio web, permitiendo modificar una parte sin afectar a la otra. Esto los convierte en la solución idónea para la creación de aplicaciones multidispositivo con acceso a contenidos web.

A continuación se enumeran otras de las ventajas del uso de un sistema gestor de contenidos frente al desarrollo tradicional de un sitio web:

- Escalabilidad y mantenibilidad: es más sencillo ampliar el sistema y mantenerlo independizando los datos de la parte gráfica.
- Consistencia de la web: se puede aplicar un mismo estilo en todas las páginas, lo que aporta un aspecto más profesional.
- Se puede controlar el acceso al contenido dependiendo de los permisos o el tipo de dispositivo que quiere acceder a la información.
- Aunque hay servicios que transforman sitios web para su visualización en smartphones, las webs convertidas suelen ser de difícil lectura y no satisfacen las necesidades de los usuarios.

Algunos de los CMS más populares podrían ser:

- **Wordpress** (http://wordpress.org/)**:** Muy conocido para la creación de blogs, es el CMS más utilizado y el mejor valorado. Gratuito.
- **Vignette** (http://www.vignette.com/)**:** Fue el primer CMS comercial que apareció en el mercado.
- **Drupal** (http://drupal.org/)**:** Gratuito y open source, ofrece la posibilidad posibilidad de utilizar distintas bases de datos. [1]

2.2. OBJETIVOS DEL PROYECTO

Creación de un CMS que gestione información de la Universidad

Debido a la creciente variedad de dispositivos desde los cuales los usuarios acceden no solo a páginas web, sino a todo tipo de aplicaciones y servicios, surge la idea de crear un sistema que permita adaptarse rápidamente a cada uno de ellos sin tener que desarrollar soluciones específicas en cada caso. Por lo tanto, se tratará de crear un CMS que, al tener un importante nivel de abstracción entre sus capas, supla esta carencia.

Implementación para iPhone

Para comprobar la viabilidad del sistema desarrollado, se creará una capa de presentación específica para iPhone.

Integración de diferentes tecnologías

Puesto que el proyecto se desarrollará utilizando una arquitectura de tres capas, y cada una de ellas se desarrollará en entornos e incluso sistemas operativos diferentes, el objetivo es cohesionar las diferentes tecnologías de modo que la transición entre las mismas sea fluida y eficaz.

Escalabilidad

Ya que en su comienzo tan solo contará con un número limitado de los múltiples servicios que la UC ofrece a la comunidad universitaria, el gestor de contenidos deberá ser escalable para poder adaptarse a un mayor número de funcionalidades o de prestaciones futuras.

Ergonomía

Además de tener contenidos útiles, es necesario que estos se presenten de forma atractiva para el usuario, algo fundamental para que la aplicación tenga aceptación y sea utilizada con asiduidad. Reducir en la medida de lo posible los tiempos de carga también debería ser prioritario.

Multidispositivo

Mi intención es crear un sistema que deje la puerta abierta a su implantación en todo tipo de dispositivos de la manera más sencilla posible.

2.3. ESTRUCTURA DEL DOCUMENTO

Este documento contará con las siguientes secciones, así como de tres apéndices que incluyen el código utilizado:

- **Diseño:** Donde se hablará de la arquitectura que se ha planteado para el proyecto, y de la planificación a distintos niveles que se ha realizado antes de proceder a su implementación, así como de las herramientas utilizadas.
- **Implementación:** Detallando el proceso que se ha seguido en el desarrollo del proyecto y las utilidades en las que se ha apoyado el mismo.
- **Puesta en marcha:** Medidas adicionales que se han tenido que adoptar para poner en funcionamiento el proyecto desarrollado.
- **Resultados obtenidos:** Explicando el funcionamiento de la aplicación final.
- **Conclusión:** Impresiones tras la finalización del proyecto.
- **Posibles mejoras:** Trabajos futuros que se podrían incorporar al proyecto.
- **Referencias:** Fuentes utilizadas en el desarrollo de este documento.

3. DISEÑO

3.1. ARQUITECTURA DEL SISTEMA

Se ha optado por un desarrollo siguiendo un estilo de programación en 3-capas, puesto que ofrece múltiples ventajas a la hora de diseñar una aplicación de las características deseadas, tales como:

- Fácil escalabilidad
- Independencia de la plataforma
- Facilidad de actualización

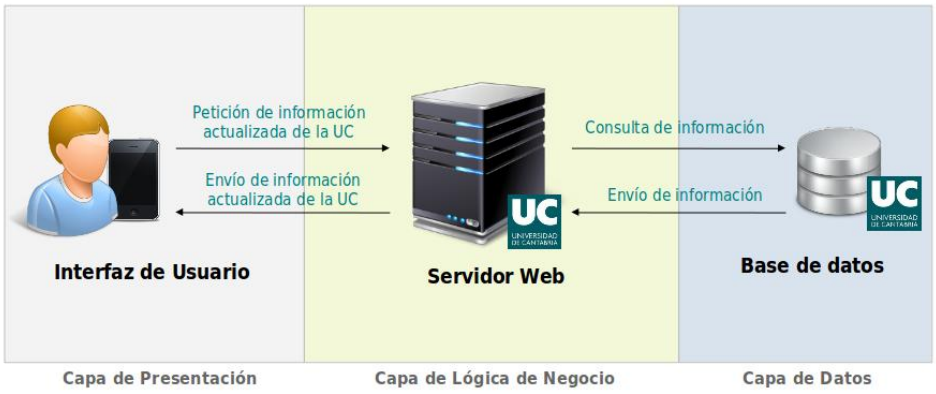


Figura 1 – Modelo 3-Capas

Como su propio nombre indica, el modelo consta de tres capas diferenciadas con las siguientes características:

- **Capa de Datos:** Donde reside la base de datos que contiene toda la información necesaria para la aplicación.
- **Lógica de Negocio:** Forma un puente entre la primera y la tercera capa. Estará formada por servicios web que reciban peticiones de la capa de presentación y accedan a la base de datos para satisfacerlas.
- **Capa de presentación:** Esta capa es la interfaz de usuario. En este caso, vendrá definida por el dispositivo en el que se quiere implementar.

En este caso se desarrollará la primera capa utilizando SQL Server, la segunda con C# y la tercera, al tratarse de una implementación para iPhone será programada en Objective-C.

Este tipo de arquitectura permitiría poder adaptar el CMS a distintos dispositivos, simplemente desarrollando una capa de presentación específica para los mismos, ya que tanto la capa de datos como la lógica de negocio serían comunes y, por lo tanto, totalmente reutilizables.

3.2. HERRAMIENTAS

3.2.1. Microsoft Excel 2007

Como paso previo a la creación de la base de datos, se realiza la planificación de la misma en una sencilla hoja de cálculo (Microsoft Excel 2007 en este caso) para, una vez definidas las diferentes tablas y campos de la misma, implementarla en SQL Server 200 R2.

3.2.2. Dia Diagram Editor

Para poder realizar los distintos diagramas de flujo que explican el funcionamiento de la aplicación, se ha utilizado este software libre que surge como alternativa al popular Microsoft Visio y que, con sus numerosas opciones de exportación, supone una gran herramienta para llevar a cabo este tipo de tareas.

3.2.3. Balsamiq Mockups

Esta útil herramienta de diseño gráfico permite, desde el navegador web, crear bocetos con multitud de recursos y opciones de personalización. De este modo, se podrá diseñar una vista preliminar de la interfaz de usuario de forma rápida e intuitiva.

3.3. DISEÑO DE LA BASE DE DATOS

En esta base de datos se almacenarán los elementos que compondrán las distintas páginas de la aplicación.

Así pues, tal y como se puede ver en la tabla "Plantillas", en este ejemplo se han definido plantillas que podrían agruparse en tres tipos:

- **Ficha**

Este tipo de plantilla engloba, como su propio nombre indica, todas las que, en general, estén destinadas a ofrecer datos específicos sobre personas, centros o departamentos. En general estarán estructuradas en diferentes campos acompañados de una o varias imágenes. En este ejemplo, a este tipo pertenecerían las plantillas **Vicerrectorado**, **Departamento** y **Centro**.

- **Texto**

Las plantillas de este grupo están formadas principalmente por bloques de texto acompañados de una o varias imágenes. Al contrario que ocurre en el tipo Ficha, en este caso las plantillas no serán, en general, específicas para un solo uso, sino que pueden utilizarse en distintos casos, como pueden ser mostrar noticias, información sobre la universidad, etc… Así pues, en este ejemplo se ha definido genéricamente la plantilla **TextoHTML**, donde se almacenarán texto enriquecido e imágenes en código HTML.

- **Mapa**

En esta categoría hemos definido la plantilla **Mapas** como una página en la que se muestran ubicaciones previamente definidas en Google Maps.

Figura 2 – Tabla "Plantillas"

Como se puede observar en la imagen, a cada plantilla se le ha asignado un ID de plantilla que servirá para identificar más adelante en qué plantilla está basada una determinada página, y por tanto qué elementos espera recibir para mostrarla.

A continuación, en la tabla "Elementos", hemos definido los elementos mencionados.

Elem_Plantilla	Nombre	Tipo	ID_Plantilla	Contenido	Múltiple	Padre	Orden	Etiqueta	Opcional
1	Nombre_VR	Texto	1	Sí	No				No
2	Persona_VR	Texto	1	Sí	No	1	1		Sí
3	Foto_VR	Imagen	1	Sí	No	1	3		Sí
4	Contacto_VR	Encabezado	1	No	No	1	2	Contacto	Sí
5	Email_VR	Email	1	Sí	No	4	1	E-mail:	Sí
6	Telefono_VR	Teléfono	1	Sí	No	4	2	Teléfono:	Sí
7	Título_HTML	Texto	5	Sí	No				No
8	Cuerpo_HTML	htmlContent	5	Sí	No	7			No
10	Nombre_Centro	Texto	3	Sí	No				No
11	Telefono_Centro	Teléfono	3	Sí	No	10	1	Teléfono:	Sí
12	Fax_Centro	Teléfono	3	Sí	No	10	2	Fax:	Sí
13	Email_Centro	Email	3	Sí	No	10	3	E-mail:	Sí
14	Mapa_Centro	Mapa	3	Sí	No	10	4		Sí
15	Foto_Centro	Imagen	3	Sí	No	10	6		Sí
16	Nombre_Dep	Texto	2	Sí	No				No
17	Telefono_Dep	Teléfono	2	Sí	No	16	1	Teléfono:	Sí
18	Fax_Dep	Teléfono	2	Sí	No	16	2	Fax:	Sí
19	Email_Dep	Email	2	Sí	No	16	3	E-mail:	Sí
20	Foto_Dep	Imagen	2	Sí	No	16	6		Sí
21	Ubica_Dep	Texto	2	Sí	No	16	4	Ubicación:	Sí
22	Otros_VR	Encabezado	1	No	No	1	5	Otras Personas	Sí
23	Nombre_Otros_VR	Texto	1	Sí	Sí	22			Sí
24	Cargo_Otros_VR	Texto	1	Sí	Sí	23	2		Sí
25	Telf_Otros_VR	Teléfono	1	Sí	Sí	23	1		Sí
26	Mapa_VR	Mapa	1	Sí	No	4	5		Sí

Figura 3 – Muestra de la tabla "Elementos"

Sin pasar a desglosar cada elemento individualmente sí que conviene explicar las distintas columnas que forman esta tabla:

- **Elem_Plantilla:** Es el ID único de un elemento en concreto.
- **Nombre:** Simplemente indica lo que representa un elemento concreto con fines organizativos.
- **Tipo:** Este parámetro indica cómo se espera que sea el dato asociado a un elemento concreto, y por tanto qué tratamiento deberá recibir.
- **ID_Plantilla:** Tal y como se ha comentado anteriormente, con el ID de plantilla se indica a qué plantilla corresponde ese elemento en concreto.
- **Contenido:** Este campo indica si el elemento tiene contenido "dinámico", en cuyo caso deberá tener un dato asociado, o si por el contrario es "estático" (es decir, un valor fijo).
- **Múltiple:** Aquí se comprueba si un determinado elemento permite multiplicidad dentro de una misma página. Por ejemplo, un centro no puede tener dos "nombres de centro", pero un vicerrectorado puede tener más de una persona asociada al mismo. La prioridad entre estos elementos iguales se establece más adelante en la tabla de "Ordenación de Datos".
- **Padre:** En este parámetro se comprueba si el elemento en cuestión depende de otros y si es así, de cuáles. Esto permite hacer asociaciones entre elementos (por ejemplo, el número de teléfono o la dirección de correo electrónico de una persona, dependerán precisamente de quién sea esa persona).
- **Orden:** Como su nombre indica, muestra el orden en el que deben ser mostrados los diferentes elementos dentro de una página. Además, en base a este campo se puede priorizar la presentación de unos elementos frente a otros en dispositivos que así lo requieran.
- **Etiqueta:** Algunos elementos, tales como los números de contacto o las direcciones de email, siempre se muestran precedidos de un determinado texto, que es lo que aquí se define. En el caso de los encabezados, es aquí donde se establece su contenido, ya que éste es fijo.
- **Opcional:** Este parámetro determina si el elemento en cuestión debe aparecer obligatoriamente en la página que se va a mostrar, o si por el contrario se puede prescindir de él. Actualmente tan solo tiene fines organizativos.

La siguiente tabla es "Páginas", donde se han definido varias páginas (cada una de ellas con un identificador único: **ID_PAG**) a modo de ejemplo, y asociadas a las diferentes plantillas. Además, en la hoja de cálculo se ha incluido un código de colores para poder identificar fácilmente a qué plantilla pertenecen, tanto en esta tabla como en el resto.

ID_PAG	Nombre	ID_Plantilla
1	Ficha VR Cultura	1
2	Ficha VR Estudiantes	1
3	Ficha VR Espacios	1
4	Ficha VR Ordenacion	1
5	Ficha VR Investigacion	1
6	Ficha VR Internacionalizacion	1
7	Facultad Ciencias	3
8	Escuela Caminos	3
9	ETSIIT	3
10	Escuela de Enfermeria	3
11	Facultad de Derecho	3
12	Dep. Física Moderna	2
13	Dep. Ciencias Médicas	2
14	Dep. Educación	2
15	Dep. Electrónica y Computadores	2
16	Historia UC	5
17	Presentación	5
18	35 Aniversario	5
19	Campus Los Castros	6
20	Dep. Biología Molecular	2
21	Pabellón de Gobierno	3
22	Facultad de Medicina	3
23	Torrelavega	6

Figura 4 – Muestra de la tabla "Páginas"

A continuación, en la tabla "Datos Páginas" se almacenan los datos de los que se van a alimentar las páginas.

ID_DATO	ELEM_PLANTILLA	VALOR
1	1	Vicerrectorado de Cultura, Participación y Difusión
2	2	Elena Martín Latorre
3	5	vr.cultura@unican.es
4	6	942201005
5	3	http://www.unican.es/NR/rdonlyres/6B0CD85D-AFFA-4DFF-9C02-F297F3F3AA97/75255/Elena_Martin_Latorre.jpg
6	7	Historia de la UC
7	8	My first page
9	7	35 Aniversario de la UC
10	10	Facultad de Ciencias
11	11	942201411
12	12	942201402
13	13	ciencias@gestion.unican.es
14	14	43.471248,-3.801236
15	15	http://www.unican.es/NR/Shared/comp/images/default_ciencias2.jpg
16	16	Departamento de Física Moderna
17	17	942201450
18	18	942201402
19	19	fisica.moderna@unican.es

Figura 5 – Muestra de la tabla "Datos"

Por tanto, esta tabla consta de una primera columna **ID_DATO** donde se define el ID único de cada uno de los datos, otra columna **ELEM_PLANTILLA** que indica a qué ID de elemento va destinado ese dato (y, por tanto, cómo se espera que sea) y por último **VALOR,** donde queda almacenado el dato propiamente dicho.

La siguiente tabla es "Ordenación de Datos", a la que ya se había hecho mención anteriormente.

ID_ORDENACION	ID_PAG	Elem_Plantilla	ID_DATO	ORDEN
1	1	1	1	
2	1	2	2	
3	1	5	3	
4	1	6	4	
5	1	3	5	
6	16	7	6	
7	16	8	7	
10	7	10	10	
11	7	11	11	
12	7	12	12	
13	7	13	13	
14	7	14	14	
15	7	15	15	
16	12	16	16	
17	12	17	17	
18	12	18	18	
19	12	19	19	
20	12	20	30	
21	12	21	10	
22	1	23	20	1
23	1	24	21	
24	1	25	22	
25	1	23	23	2
26	1	24	24	

Figura 6 – Muestra de la tabla "Ordenación"

Aquí se indica, además del ID de ordenación correspondiente, el ID de página y el ID de elemento al que se asociará un determinado dato y, en caso de que sean múltiples elementos del mismo tipo dentro de una misma página (ver explicación del campo **Múltiple** de la tabla "Items Plantilla"), se ordenarán según la prioridad que indique la columna **ORDEN**. Además, esta tabla permite reutilizar datos que aparezcan en varias páginas sin necesidad de almacenarlos en la base de datos más de una vez, reduciendo así su tamaño y aumentando su optimización. En la imagen de ejemplo hemos reutilizado el dato con ID=10 (que como podemos consultar en la tabla "Datos Páginas" se trata del nombre "Facultad de Ciencias") puesto que la ubicación del departamento de Física Moderna (elemento 16 correspondiente a la plantilla 2, en la que se basa la página 12) es la misma que la de la propia Facultad de Ciencias (elemento 10 de la plantilla 3, en la que se basa la página 7).

Por último, la tabla "Menú" es la que da forma al menú de navegación de la aplicación. Además, hasta que una página no figure en la tabla de menú, no será visible y no se podrá acceder a ella debido a que no estará enlazada a ningún elemento de menú, lo que hace a esta disposición ideal para poder desarrollar nuevo contenido sin necesidad de publicarlo hasta que esté listo y por tanto, no afectando al funcionamiento de la aplicación.

ID_Menu	Texto	Padre	Prioridad	ID_PAG	Vers
1	Información General		1		1.0.0
2	Organización		2		1.0.0
3	Vicerrectorados	2	1		1.0.0
4	Centros	2	2		1.0.0
5	Departamentos	2	3		1.0.0
6	Facultad de Ciencias	4	21	7	1.0.0
7	Escuela de Caminos	4	20	8	1.0.1
8	Vicerrectorado de Difusión	3	1	1	1.0.1
9	Campus		4		1.0.1
10	Presentación	1	1	17	1.0.1
11	Historia de la UC	1	2	16	1.0.1
12	Dept. Física Moderna	5	1	12	1.0.1
13	Dept. Electrónica y Computadores	5	2	15	1.0.1
14	Dep. Biología Molecular	5	3	20	1.0.1
15	Pabellón de Gobierno	4	1	21	1.0.1
16	Facultad de Medicina	4	25	22	1.0.1
17	Vicerrectorado de Estudiantes	3	2	2	1.0.1
18	Vicerrectorado de Espacios	3	3	3	1.0.1
19	Avenida de los Castros	9	1	19	1.0.1
20	35 Aniversario	1	3	18	1.0.1
21	Torrelavega	9	2	23	1.0.1

Figura 7 – Muestra de la tabla “Menú”

Como se puede observar en la imagen, las columnas que conformas esta tabla son:

- **ID_Menu:** Como en anteriores tablas, se trata del identificador único de cada entrada dentro de la tabla actual.
- **Texto:** Define el texto que aparecerá en el dispositivo para esa entrada del menú.
- **Padre:** Por medio de esta columna definimos el sistema de jerarquía y la navegación del propio menú. Si una entrada no tiene padre, será mostrada en el menú inicial, mientras que si al seleccionar un elemento, mostrará sus hijos.
- **Prioridad:** Este parámetro indica en qué orden deberá presentarse la información.
- **ID_Pag:** En caso de que un elemento tenga asociado un ID de página, al seleccionarlo terminará la navegación para proceder a cargar la página correspondiente.
- **Vers:** Muestra la versión del menú a la que pertenece una entrada, siendo el mayor valor de esta columna la que se considera como versión actual del menú.

3.4. DEFINICIÓN DE SERVICIOS WEB

Los servicios web serán los encargados de recoger las peticiones de la capa de presentación, realizar las consultas necesarias en la base de datos para satisfacerlas, y devolver los resultados correspondientes.

Por ello, analizando las posibles necesidades que puede presentar la interfaz de usuario, se plantea la siguiente secuencia de funcionamiento, que deja entrever las peticiones necesarias para su correcto funcionamiento:

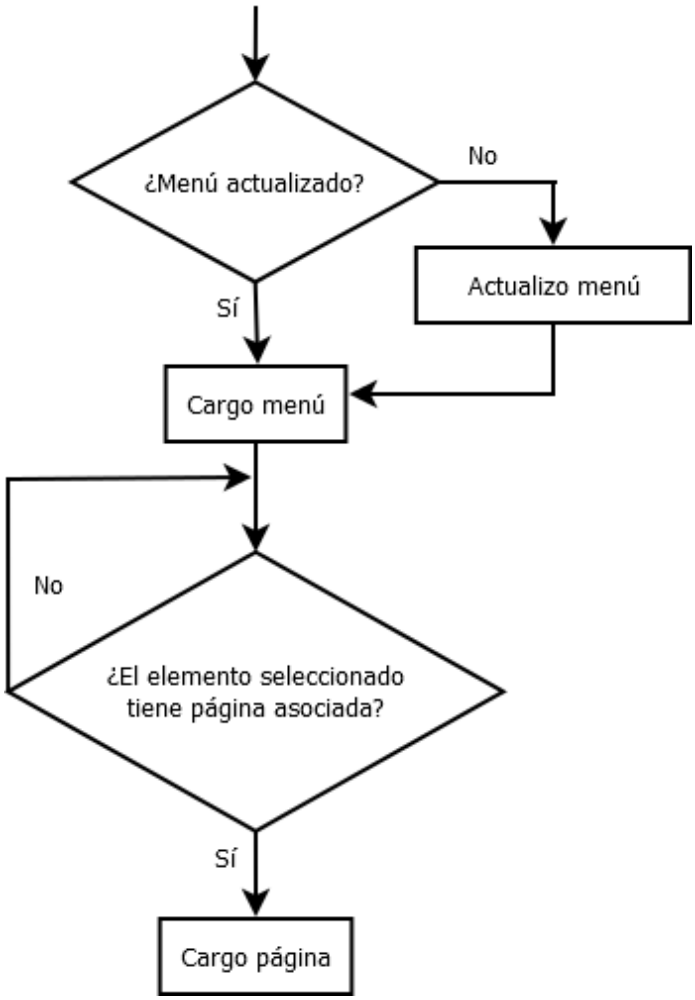


Figura 8 – Diagrama básico de funcionamiento

Así pues, planteo las siguientes funciones:

Nombre	**versionMenu**
Descripción	Comprueba si la versión actual del dispositivo es la más reciente
Parámetros de entrada	currentVersion (string)
Parámetros de salida	isUpdated (boolean)

Nombre	**getLatestMenu**
Descripción	Obtiene la última versión disponible de la tabla de menús
Parámetros de entrada	Ninguno
Parámetros de salida	XML con el contenido de la tabla Menú

Nombre	**getPage**
Descripción	Obtiene el contenido de la página solicitada para un dispositivo determinado a partir del identificador de página
Parámetros de entrada	pageID (int), device (string)
Parámetros de salida	XML con el contenido de la página solicitada

El formato elegido para almacenar los resultados ha sido XML (e**X**tensible **M**arkup **L**anguage), un lenguaje utilizado para estructurar grandes documentos y que permite compartir información entre distintos sistemas de una manera segura, fiable y fácil. [2]

3.5. DISEÑO DE LA INTERFAZ DE USUARIO

Tal y como se comenta en el apartado Herramientas, la vista previa para la implementación en el dispositivo se ha realizado utilizando la herramienta Balsamiq Mockups. Así pues, se han diseñado diferentes vistas del menú y de las páginas, en consonancia con lo almacenado en la base de datos, que servirán de referencia a la hora de realizar el desarrollo de la aplicación para iPhone.

Como ejemplo se muestran los bocetos del menú y de varios tipos de plantillas, como son las de centro, vicerrectorado o texto, mostrando la información almacenada para cada una de ellas:

Figura 9 – Diseño preliminar del menú y una ficha de centros

Figura 10 – Borrador de ficha de Vicerrectorado y de plantilla de texto

4. IMPLEMENTACIÓN

4.1. HERRAMIENTAS

4.1.1. SQL Server 2008 R2

Para la implementación de la base de datos se ha utilizado **SQL Server 2008 R2** de Microsoft. La decisión de utilizar esta herramienta en este caso venía determinada por el cliente, la Universidad de Cantabria, que actualmente utiliza este tipo de sistemas para el almacenamiento de información, haciendo por tanto que el coste de implantación del nuevo sistema de gestión de contenidos sea mínimo.

Tanto SQL Server como Oracle o MySQL son sistemas gestores de bases de datos muy potentes, bien documentados y que cuentan con un gran equipo de expertos detrás para dar soporte y resolver dudas sobre su instalación y manejo. SQL Server 2008 R2, se trata de un gestor muy robusto y escalable que permite la creación de aplicaciones con .NET Framework y Visual Studio con cualquier grado de complejidad. Aunque sólo está disponible para Windows, desde la versión de 2008 SQL Server es capaz de competir con grandes sistemas como Oracle principalmente por su facilidad de uso y por ofrecer un rendimiento cada vez más cercano a las cifras de este a un menor coste. Además incluye una serie de características, como la tecnología *Always On* de Microsoft, que aumentan la disponibilidad del sistema minimizando las pérdidas de servicio y facilitando la recuperación. [3]

Para el acceso, administración y configuración de todos los componentes de SQL Server 2008 R2 se ha utilizado **SQL Server Management Studio**, una herramienta que permite configurar, gestionar y administrar todos los componentes de la base de datos.[4]

4.1.2. Visual Studio 2010

Para el desarrollo de la lógica de negocio necesaria para comunicar la capa de datos con la de presentación, se ha utilizado el entorno de desarrollo de **Microsoft Visual Studio 2010** para Windows. Este entorno soporta numerosos lenguajes de programación como Visual C++, Visual Basic .NET o Visual C#, entre otros. [5]

En este caso se ha elegido C#, un lenguaje orientado objetos resultante de la evolución de C y C++ que incorpora algunas de las mejoras de productividad de JAVA, y que forma parte de la plataforma .NET de Microsoft.[6]

4.1.3. Xcode 4.3.2

Por último, para el desarrollo de la capa de presentación, el entorno utilizado ha sido **Xcode** de Apple para Mac OS X v10.7 (Lion) en su versión 4.3.2. Esta vez, la decisión viene impuesta por el dispositivo en el que se va a realizar la implementación, ya que el desarrollo una aplicación nativa para iOS es necesario realizarlo en lenguaje Objective-C, que tan solo ha sido estandarizado por Apple.

En cualquier caso, Xcode es una herramienta de programación muy potente y que guarda ciertas similitudes con otros entornos, como Visual Studio, mientras que tiene la enorme ventaja de que está enormemente orientado a dispositivos iOS, por lo que la integración de estos en la herramienta es total.

Como ya se ha comentado, el lenguaje en el que se llevará a cabo la aplicación será Objective-C, otro lenguaje orientado a objetos que surge a partir de C por lo que, al igual que C#, comparte mucha de la sintaxis y de las características del lenguaje original.[7]

4.2. CAPA DE DATOS (SQL Server 2008 R2)

Siguiendo el diseño de las tablas en Excel, procedo a plasmarlas en una base de datos de SQL Server 2008 R2. Para ello, creo una nueva base de datos, y mediante consultas, voy creando las diferentes tablas, con sus respectivas columnas, y las dependencias entre estas.

Como se ha podido observar en la fase de diseño, hay varias columnas que aparecen en dos o más tablas diferentes. Para reflejar esta relación, se hace uso de dos tipos de claves, las primary keys y las foreign keys.

- **Primary Keys:** Las primary keys (PK), o claves principales de una tabla, son columnas o combinaciones de columnas cuyos valores identifican de forma única cada fila de la misma. Esto hace que en estas columnas no puedan existir valores duplicados.[8]

- **Foreign Keys:** Las foreign keys (FK), son columnas o combinaciones de columnas de una tabla (tabla hija) que se refieren a la primary key de otra tabla (tabla maestra). Estas columnas tan solo aceptarán valores previamente definidos en las claves principales.[9]

Por tanto, resulta fácil identificar las columnas candidatas a ser primary key en las diferentes tablas y, en caso de que existan, las foreign keys correspondientes, quedando el diagrama de la base de datos como se puede observar en la siguiente imagen, donde las llaves representan las claves principales, y las líneas que conectan distintas tablas representan una relación PK-FK:

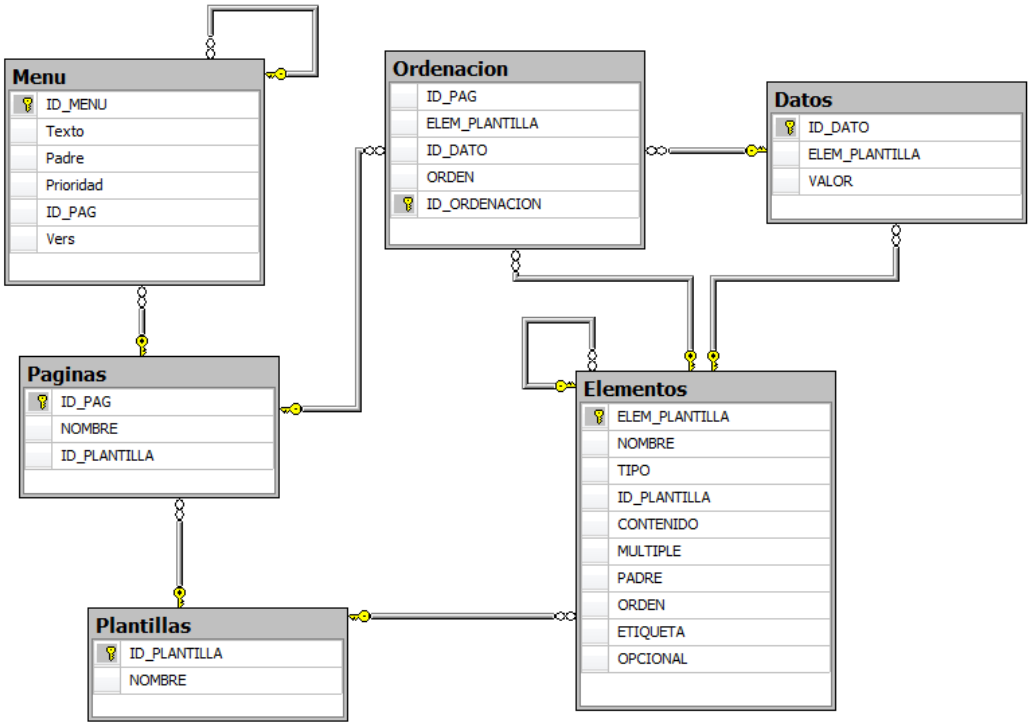


Figura 11 – Diagrama de la Base de Datos

Se puede apreciar que tanto en la tabla Menú como en la tabla Elementos existe una relación PK-FK que referencia a su propia tabla. En ambos casos, esto se debe a que la columna "Padre" es foreign key de la clave primaria de la tabla (ID_MENU y ELEM_PLANTILLA, respectivamente) puesto que tan solo debe admitir valores que existan previamente en la primary key.

A continuación paso a detallar el tipo de dato y las propiedades, si las hubiera, que he elegido las diferentes columnas de cada tabla:

PLANTILLAS			
Columna	**Tipo**	**Permite valores NULL**	**Otras propiedades**
ID_PLANTILLA	int	No	Primary Key
NOMBRE	varchar(30)	No	-
PÁGINAS			
Columna	**Tipo**	**Permite valores NULL**	**Otras propiedades**
ID_PAG	int	No	Primary Key
NOMBRE	varchar(60)	No	-
ID_PLANTILLA	int	No	Foreign Key (Plantillas)

ELEMENTOS			
Columna	**Tipo**	**Permite valores NULL**	**Otras propiedades**
ELEM_PLANTILLA	int	No	Primary Key
NOMBRE	varchar(30)	No	-
TIPO	varchar(20)	No	-
ID_PLANTILLA	int	No	Foreign Key (Plantillas)
CONTENIDO	tinyint	No	-
MÚLTIPLE	tinyint	No	-
PADRE	int	Sí	Foreign Key (Elementos)
ORDEN	int	Sí	-
ETIQUETA	varchar(30)	Sí	-
OPCIONAL	tinyint	No	-
DATOS			
Columna	**Tipo**	**Permite valores NULL**	**Otras propiedades**
ID_DATO	int	No	Primary Key
ELEM_PLANTILLA	int	No	Foreign Key (Elementos)
VALOR	varchar(max)	No	-
ORDENACIÓN			
Columna	**Tipo**	**Permite valores NULL**	**Otras propiedades**
ID_ORDENACION	int	No	Primary Key
ID_PAG	int	No	Foreign Key (Páginas)
ELEM_PLANTILLA	int	No	Foreign Key (Elementos)
ID_DATO	int	No	Foreign Key (Datos)
ORDEN	int	Sí	-
MENÚ			
Columna	**Tipo**	**Permite valores NULL**	**Otras propiedades**
ID_MENU	int	No	Primary Key
TEXTO	varchar(max)	No	-
PADRE	int	Sí	Foreign Key (Menú)

PRIORIDAD	int	No	-
ID_PAG	int	Sí	Foreign Key (Páginas)
VERS	varchar(12)	No	-

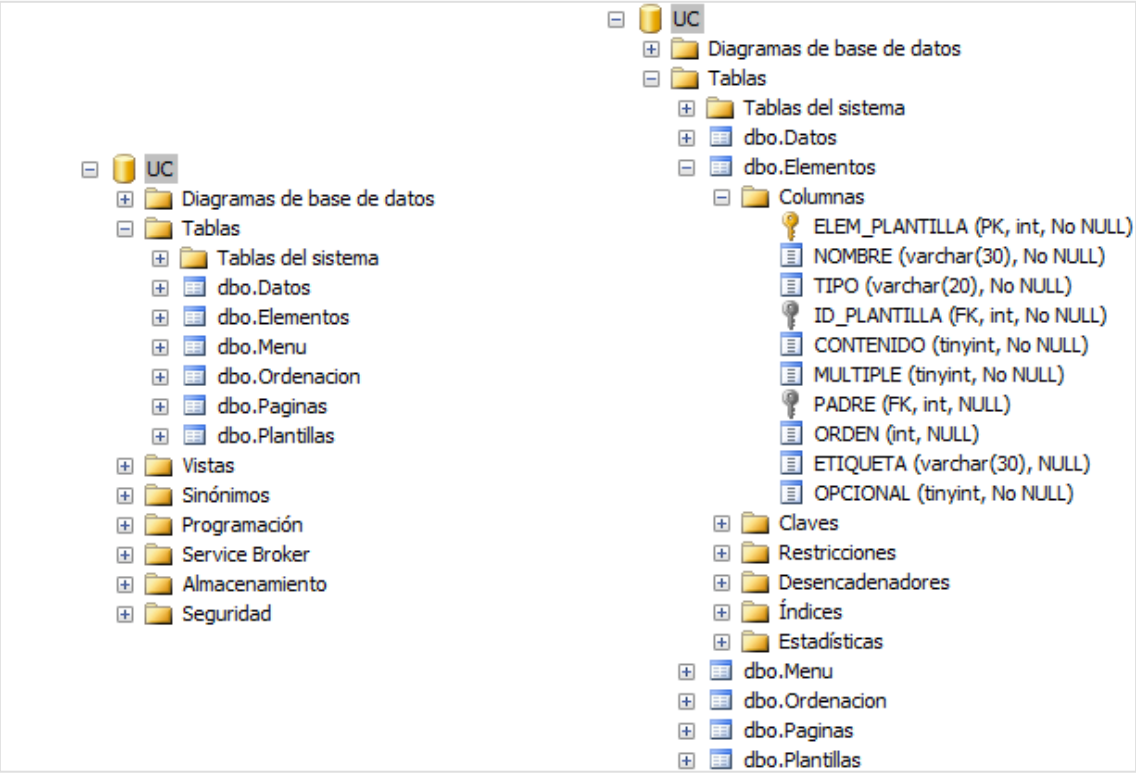


Figura 12 – Tablas en la base de datos y columnas de la tabla "Elementos"

Tras definir cada una de las tablas con sus correspondientes columnas, se procede a volcar los datos en las mismas mediante consultas.

```
INSERT INTO Datos
(ELEM_PLANTILLA,VALOR)
VALUES
(1,'Vicerrectorado de Estudiantes, Empleabilidad y Emprendimiento'),
(2,'Rafael Pedro Torres Jiménez'),
(3,'http://www.unican.es/NR/rdonlyres/74145/Torres.jpg'),
(5,'vr.estudiantes@unican.es'),
(6,'942201009'),
(28,'http://www.unican.es/Vicerrectorados/estudiantes/'),
(33,'942202274'),
(34,'Vicerrector.

Departamento de Ingeniería de Comunicaciones.');
```

Figura 13 – Ejemplo de consulta de introducción de datos

Una vez introducidos, los datos quedarían de la siguiente forma:

	ELEM_PLANTILLA	NOMBRE	TIPO	ID_PLANTILLA	CONTENIDO	MULTIPLE	PADRE	ORDEN	ETIQUETA	OPCIONAL
1	1	Nombre_VR	Texto	1	1	0	NULL	NULL	NULL	0
2	2	Persona_VR	Texto	1	1	0	1	1	NULL	1
3	3	Foto_VR	Imagen	1	1	0	1	3	NULL	1
4	4	Contacto_VR	Encabezado	1	0	0	1	2	Contacto	1
5	5	Email_VR	Email	1	1	0	4	1	E-mail:	1
6	6	Telefono_VR	Telefono	1	1	0	4	2	Teléfono	1
7	7	Titulo_TI	Texto	5	1	0	NULL	NULL	NULL	0
8	8	Texto_TI	Texto	5	1	1	7	NULL	NULL	0
9	9	Imagen_TI	Imagen	5	1	1	8	NULL	NULL	1
10	10	Nombre_Centro	Texto	3	1	0	NULL	NULL	NULL	0
11	11	Telefono_Centro	Telefono	3	1	0	10	1	Teléfono:	1
12	12	Fax_Centro	Telefono	3	1	0	10	3	Fax:	1

Figura 14 – Muestra de la tabla “Elementos” en la Base de Datos

4.3. LÓGICA DE NEGOCIO (Visual Studio 2010)

4.3.1. Entity Framework

Para hacer uso de la información almacenada en la base de datos desde nuestro programa en Visual Studio, en primer lugar trato de utilizar la clase SqlConnection, pero presenta varias deficiencias, ya que no permite herencia ni presenta ningún nivel de abstracción, por lo que cualquier cambio realizado a posteriori en la base de datos no se vería reflejado, y en consecuencia tendríamos problemas en las conexiones.

Así pues, procedo a crear lo que se denomina Entity Framework, que mapea la estructura y las relaciones de la base de datos en un modelo conceptual que permitirá manejar la información con mayor facilidad.

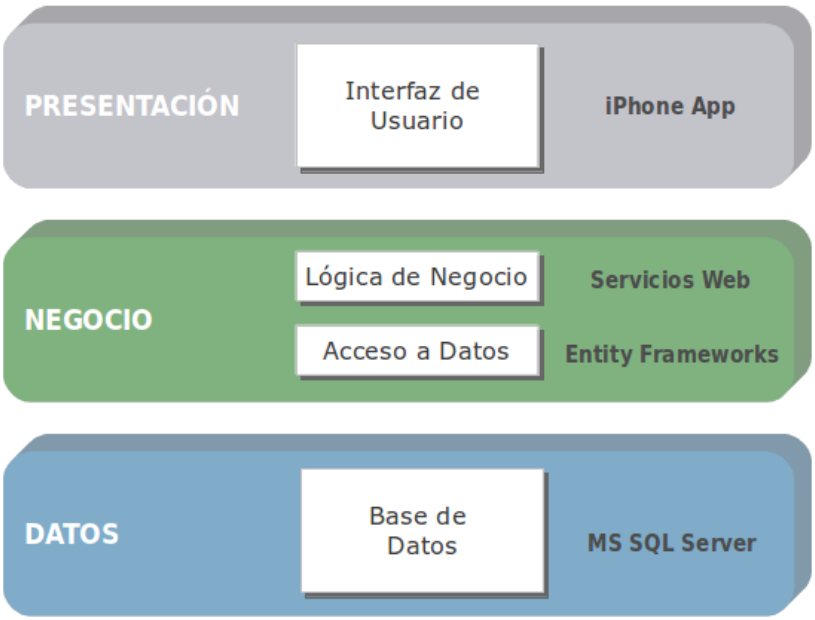


Figura 15 – Arquitectura tras la inclusión de Entity Frameworks

Entity Framework consta de un conjunto de APIs de acceso a datos para Microsoft .NET y viene incluido en Visual Studio 2010, lo que lo convierte en una solución idónea.

Tras realizar dicho mapeo, nuestro modelo de datos queda de la siguiente manera:

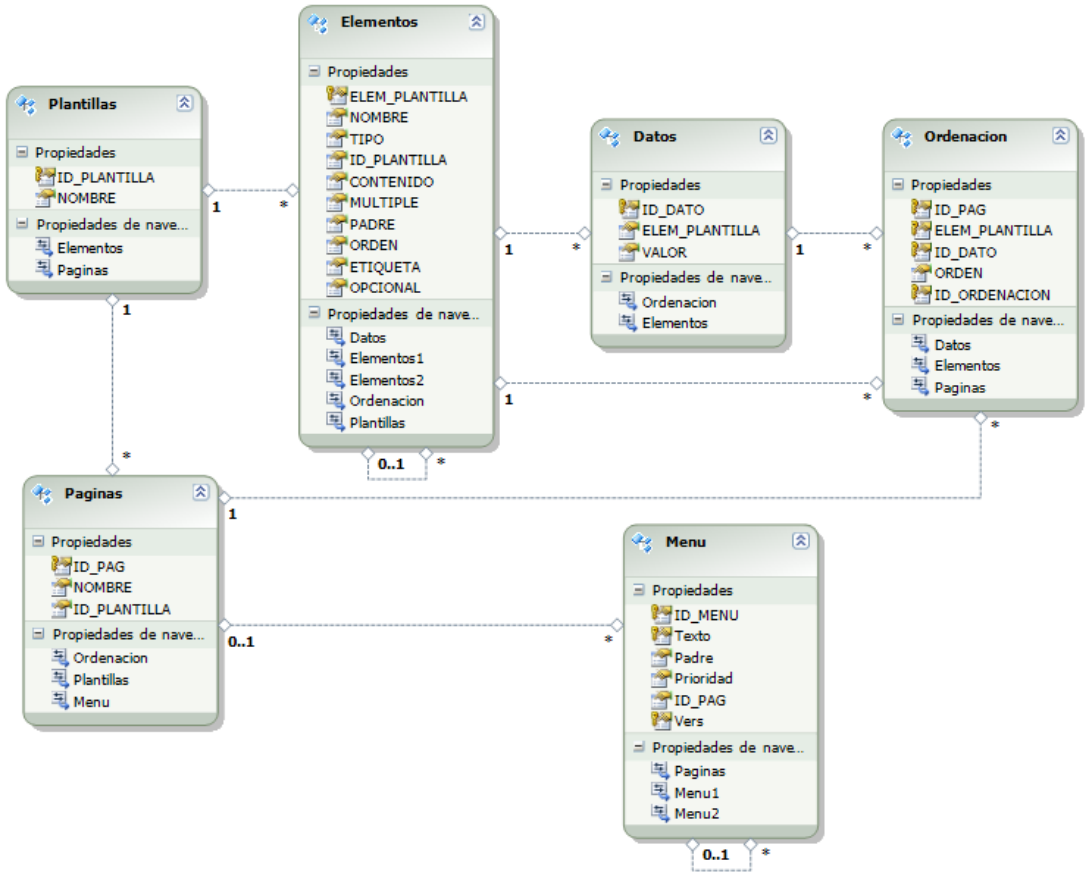


Figura 16 – Modelo de datos de nuestro Entity Framework

4.3.2. Servicios Web

Tras crear el modelo de acceso a datos, el siguiente paso será crear los servicios web necesarios para hacer llegar al dispositivo la información que requiera en cada momento.

En este caso se han creado tres servicios web con las funciones definidas originalmente (versionMenu, getLatestMenu y getPage) con pequeños matices, y cuyo funcionamiento se describe a continuación.

4.3.2.1. versionMenu

Nombre	**versionMenu**
Descripción	Indica la versión actual de la tabla de menú

Parámetros de entrada	Ninguno
Parámetros de salida	maximo (string)

En un principio pensada para recibir un parámetro, realizar por sí misma la comparación, y devolver el resultado, la función versionMenu se ha modificado para, simplemente, devolver el valor de la versión de menú en un string y dejar que el dispositivo sea quien compare dicho valor con el que actualmente tiene almacenado. Este cambio se ha realizado solamente por cuestiones de rendimiento.

```
<string xmlns="http://tempuri.org/">1.0.1</string>
```

Figura 17 – Ejemplo de respuesta de versionMenu

Cabe recordar que la versión actual de la base de datos se obtiene calculando el máximo de la columna Vers de la tabla Menú.

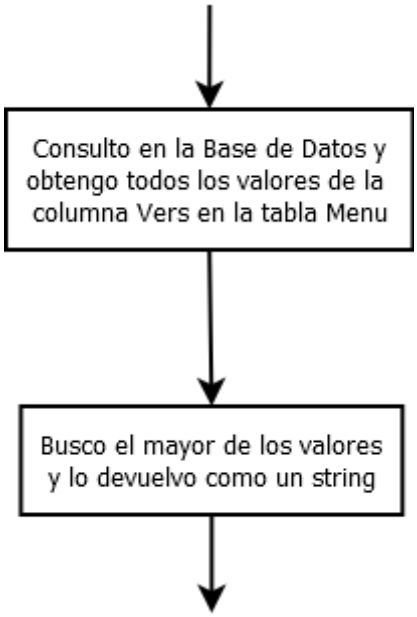


Figura 18 – Diagrama de flujo de versionMenu

4.3.2.2. getLatestMenu

Nombre	**getLatestMenu**
Descripción	Obtiene la última versión disponible de la tabla de menús
Parámetros de entrada	Ninguno
Parámetros de salida	XML con el contenido de la tabla Menú

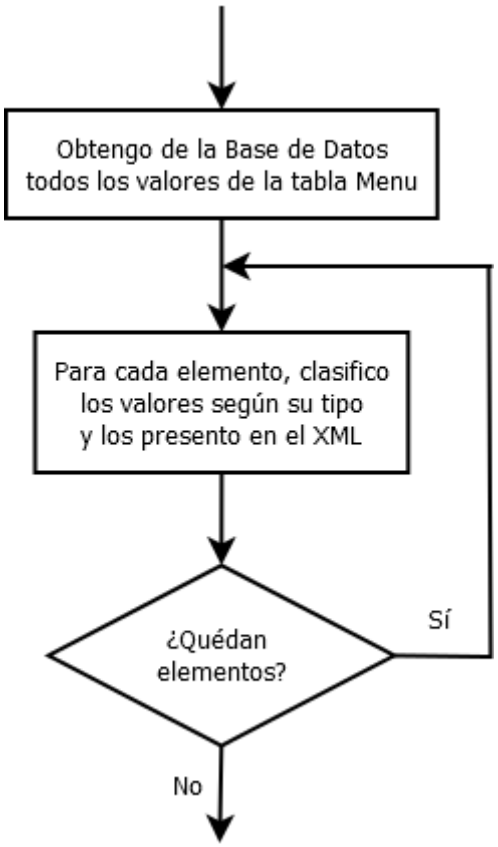
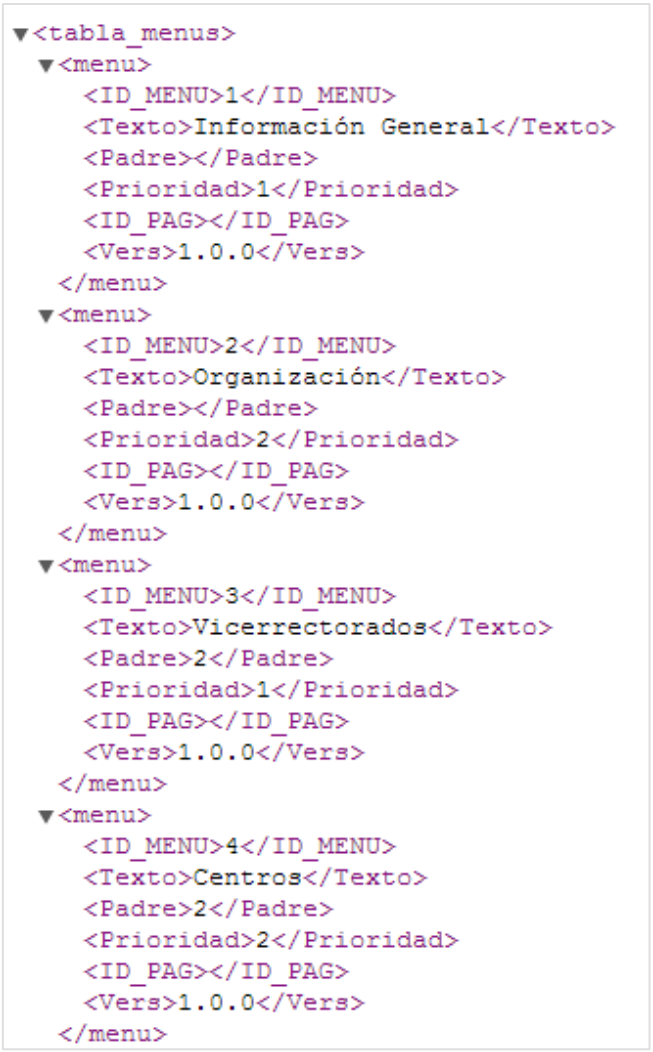


Figura 19 – Diagrama de flujo de getLatestMenu

Cuando es invocada, la función getLatestMenu vuelca la última versión de la tabla Menú en la memoria del dispositivo. Para ello, toma toda la información de la tabla, la agrupa por filas y para cada uno de sus campos se añaden etiquetas que ayuden a determinar a qué columna corresponde cada dato. Esto se hace de este modo para tener organizada la información, ya que el servidor no tiene por qué recoger los datos siguiendo siempre un mismo orden.



```
▼<tabla_menus>
  ▼<menu>
     <ID_MENU>1</ID_MENU>
     <Texto>Información General</Texto>
     <Padre></Padre>
     <Prioridad>1</Prioridad>
     <ID_PAG></ID_PAG>
     <Vers>1.0.0</Vers>
   </menu>
  ▼<menu>
     <ID_MENU>2</ID_MENU>
     <Texto>Organización</Texto>
     <Padre></Padre>
     <Prioridad>2</Prioridad>
     <ID_PAG></ID_PAG>
     <Vers>1.0.0</Vers>
   </menu>
  ▼<menu>
     <ID_MENU>3</ID_MENU>
     <Texto>Vicerrectorados</Texto>
     <Padre>2</Padre>
     <Prioridad>1</Prioridad>
     <ID_PAG></ID_PAG>
     <Vers>1.0.0</Vers>
   </menu>
  ▼<menu>
     <ID_MENU>4</ID_MENU>
     <Texto>Centros</Texto>
     <Padre>2</Padre>
     <Prioridad>2</Prioridad>
     <ID_PAG></ID_PAG>
     <Vers>1.0.0</Vers>
   </menu>
```


Figura 20 – Muestra de la respuesta de getLatestMenu

4.3.2.3. getPage

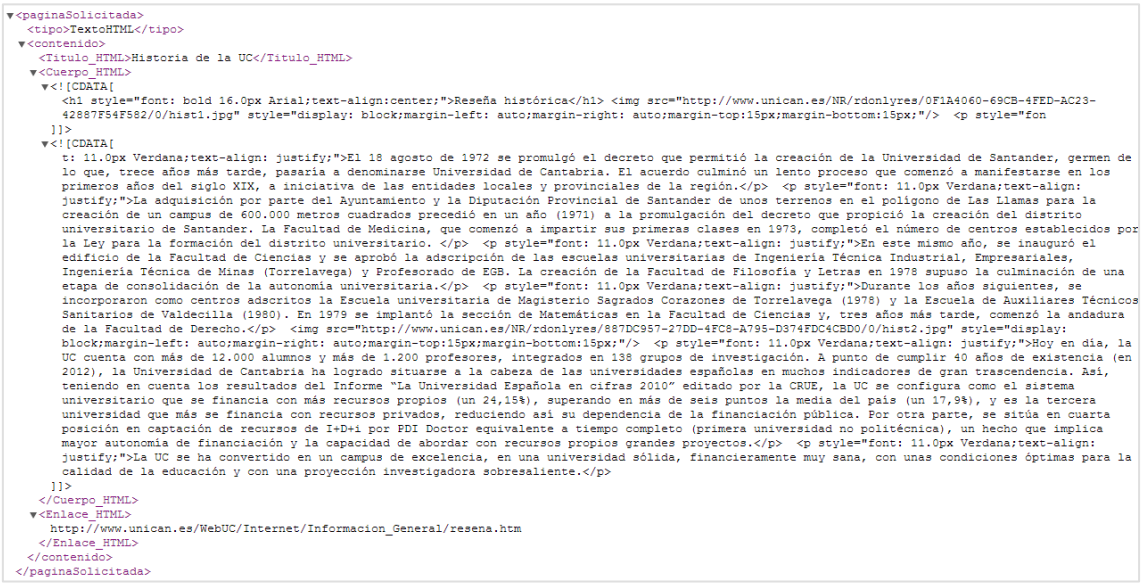
Nombre	**getPage**
Descripción	Obtiene el contenido de la página solicitada para un dispositivo determinado a partir del identificador de página
Parámetros de entrada	pageID (int)
Parámetros de salida	XML con el contenido de la página solicitada

La función getPage es la encargada de cargar cualquier tipo de página que la aplicación requiera, por lo que debe estar preparada para devolver páginas de cualquiera de los tipos de plantilla existentes. Esto hace que tenga un nivel de complejidad tremendamente superior a las dos funciones ya vistas, e incluso necesite de dos

subrutinas diferentes (getContenidoMapa y getContenidoPage) dependiendo de si la página solicitada es de tipo Mapa o de cualquiera de los demás.

El motivo por el que se hace esta distinción es que en las páginas del tipo Mapa necesito obtener varios elementos del mismo tipo y emparejarlos de manera que a cada nombre se le asocien sus coordenadas correspondientes.

Dentro de getContenidoPage también hay un caso particular, y es que las páginas del tipo TextoHTML, al tener en su código etiquetas que podrían llevar a confusión a la hora de parsear (descrifrar) la información del archivo XML, han de tener los datos del tipo Cuerpo_HTML contenidos en una sección CDATA, lo que hace que se almacenen "en bruto" y no se interpreten como parte de la estructura XML.



```
▼<paginaSolicitada>
  <tipo>TextoHTML</tipo>
 ▼<contenido>
   <Titulo_HTML>Historia de la UC</Titulo_HTML>
  ▼<Cuerpo_HTML>
   ▼<![CDATA[
      <h1 style="font: bold 16.0px Arial;text-align:center;">Reseña histórica</h1> <img src="http://www.unican.es/NR/rdonlyres/0F1A4060-69CB-4FED-AC23-
      42887F54F582/0/hist1.jpg" style="display: block;margin-left: auto;margin-right: auto;margin-top:15px;margin-bottom:15px;"/>  <p style="fon
    ]]>
   ▼<![CDATA[
      t: 11.0px Verdana;text-align: justify;">El 18 agosto de 1972 se promulgó el decreto que permitió la creación de la Universidad de Santander, germen de
      lo que, trece años más tarde, pasaría a denominarse Universidad de Cantabria. El acuerdo culminó un lento proceso que comenzó a manifestarse en los
      primeros años del siglo XIX, a iniciativa de las entidades locales y provinciales de la región.</p>  <p style="font: 11.0px Verdana;text-align:
      justify;">La adquisición por parte del Ayuntamiento y la Diputación Provincial de Santander de unos terrenos en el polígono de Las Llamas para la
      creación de un campus de 600.000 metros cuadrados precedió en un año (1971) a la promulgación del decreto que propició la creación del distrito
      universitario de Santander. La Facultad de Medicina, que comenzó a impartir sus primeras clases en 1973, completó el número de centros establecidos por
      la Ley para la formación del distrito universitario. </p>  <p style="font: 11.0px Verdana;text-align: justify;">En este mismo año, se inauguró el
      edificio de la Facultad de Ciencias y se aprobó la adscripción de las escuelas universitarias de Ingeniería Técnica Industrial, Empresariales,
      Ingeniería Técnica de Minas (Torrelavega) y Profesorado de EGB. La creación de la Facultad de Filosofía y Letras en 1978 supuso la culminación de una
      etapa de consolidación de la autonomía universitaria.</p>  <p style="font: 11.0px Verdana;text-align: justify;">Durante los años siguientes, se
      incorporaron como centros adscritos la Escuela universitaria de Magisterio Sagrados Corazones de Torrelavega (1978) y la Escuela de Auxiliares Técnicos
      Sanitarios de Valdecilla (1980). En 1979 se implantó la sección de Matemáticas en la Facultad de Ciencias y, tres años más tarde, comenzó la andadura
      de la Facultad de Derecho.</p>  <img src="http://www.unican.es/NR/rdonlyres/887DC957-27DD-4FC8-A795-D374FDC4CBD0/0/hist2.jpg" style="display:
      block;margin-left: auto;margin-right: auto;margin-top:15px;margin-bottom:15px;"/>  <p style="font: 11.0px Verdana;text-align: justify;">Hoy en día, la
      UC cuenta con más de 12.000 alumnos y más de 1.200 profesores, integrados en 138 grupos de investigación. A punto de cumplir 40 años de existencia (en
      2012), la Universidad de Cantabria ha logrado situarse a la cabeza de las universidades españolas en muchos indicadores de gran trascendencia. Así,
      teniendo en cuenta los resultados del Informe "La Universidad Española en cifras 2010" editado por la CRUE, la UC se configura como el sistema
      universitario que se financia con más recursos propios (un 24,15%), superando en más de seis puntos la media del país (un 17,9%), y es la tercera
      universidad que más se financia con recursos privados, reduciendo así su dependencia de la financiación pública. Por otra parte, se sitúa en cuarta
      posición en captación de recursos de I+D+i por PDI Doctor equivalente a tiempo completo (primera universidad no politécnica), un hecho que implica
      mayor autonomía de financiación y la capacidad de abordar con recursos propios grandes proyectos.</p>  <p style="font: 11.0px Verdana;text-align:
      justify;">La UC se ha convertido en un campus de excelencia, en una universidad sólida, financieramente muy sana, con unas condiciones óptimas para la
      calidad de la educación y con una proyección investigadora sobresaliente.</p>
    ]]>
  </Cuerpo_HTML>
 ▼<Enlace_HTML>
    http://www.unican.es/WebUC/Internet/Informacion_General/resena.htm
  </Enlace_HTML>
 </contenido>
</paginaSolicitada>
```


Figura 21 – Muestra de la respuesta de getPage para página tipo TextoHTML

Además, en previsión a una futura implantación en otros dispositivos, getPage debe establecer hasta qué nivel de datos debe cargar para cada uno de ellos, ya que por ejemplo en este caso (iPhone) en las páginas de Vicerrectorado tan solo se mostrará al Vicerrector, pero en casos de dispositivos de mayor resolución podrían mostrarse un número de personas más elevado (cuyos campos ya se han definido en la base de datos pero que en esta implementación no se muestran, sirviendo como ejemplo de este filtrado).

A continuación podemos observar tanto su diagrama de funcionamiento como su respuesta ante peticiones de páginas pertenecientes a los distintos tipos de plantilla.

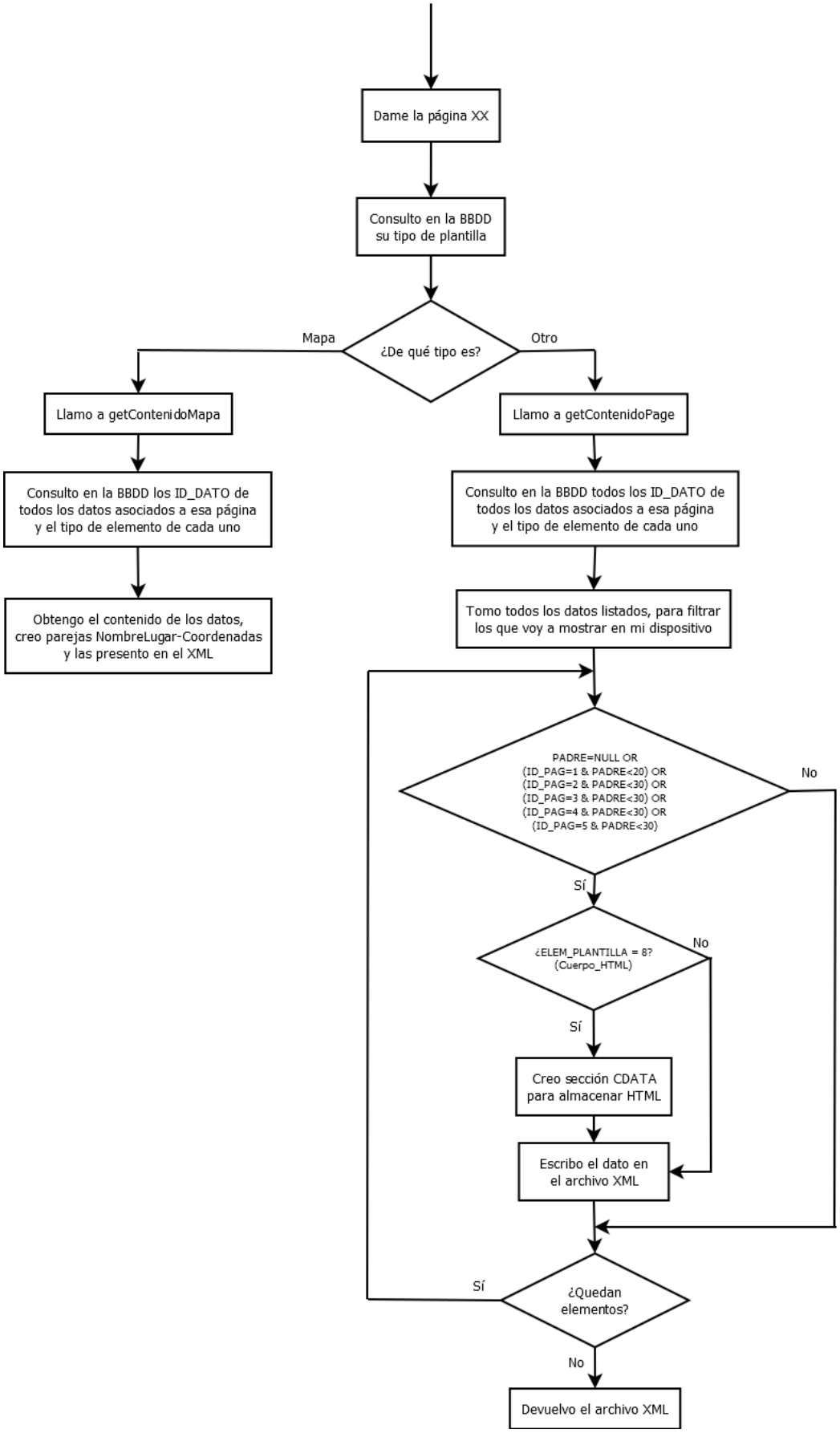


Figura 22 – Diagrama de Flujo de la función getPage

```
▼<paginaSolicitada>
  <tipo>Ficha</tipo>
 ▼<contenido>
  ▼<Nombre_VR>
     Vicerrectorado de Cultura, Participación y Difusión
   </Nombre_VR>
   <Persona_VR>Elena Martín Latorre</Persona_VR>
   <Email_VR>vr.cultura@unican.es</Email_VR>
   <Telefono_VR>942201005</Telefono_VR>
  ▼<Foto_VR>
     http://www.unican.es/NR/rdonlyres/6B0CD85D-AFFA-4DFF-9C02-F297F3F3AA97/75255/Elena_Martin_Latorre.jpg
   </Foto_VR>
   <Mapa_VR>43.470451,-3.805289</Mapa_VR>
   <Enlace_VR>http://www.unican.es/Vicerrectorados/difusion/</Enlace_VR>
   <Ubica_VR>Pabellón de Gobierno</Ubica_VR>
   <Fax_VR>942202040</Fax_VR>
  ▼<Cargo_VR>
     Vicerrectora Profesora Titular de Universidad Departamento de Geografía, Urbanismo y Ordenación del territorio.
   </Cargo_VR>
  </contenido>
 </paginaSolicitada>
```


Figura 23 – Muestra de la respuesta de getPage para página tipo Vicerrectorado


```
▼<paginaSolicitada>
   <tipo>Ficha</tipo>
  ▼<contenido>
     <Nombre_Centro>Facultad de Ciencias</Nombre_Centro>
     <Telefono_Centro>942201411</Telefono_Centro>
     <Fax_Centro>942201402</Fax_Centro>
     <Email_Centro>ciencias@gestion.unican.es</Email_Centro>
     <Mapa_Centro>43.471248,-3.801236</Mapa_Centro>
    ▼<Foto_Centro>
       http://www.unican.es/NR/Shared/comp/images/default_ciencias2.jpg
     </Foto_Centro>
     <Enlace_Centro>http://www.unican.es/Centros/ciencias/</Enlace_Centro>
   </contenido>
 </paginaSolicitada>
```


Figura 24 – Muestra de la respuesta de getPage para página tipo Centro


```
▼<paginaSolicitada>
   <tipo>Mapa</tipo>
  ▼<localizaciones>
   ▼<loc>
      <Nombre_Mapa>Escuela de Minas</Nombre_Mapa>
      <Coord_Mapa>43.337368,-4.051307</Coord_Mapa>
    </loc>
   ▼<loc>
      <Nombre_Mapa>Escuela de Fisioterapia</Nombre_Mapa>
      <Coord_Mapa>43.336752,-4.04909</Coord_Mapa>
    </loc>
   </localizaciones>
 </paginaSolicitada>
```


Figura 25 – Muestra de la respuesta de getPage para página tipo Mapa

4.4. INTERFAZ DE USUARIO (Xcode 4.3.2)

4.4.1. Frameworks utilizados

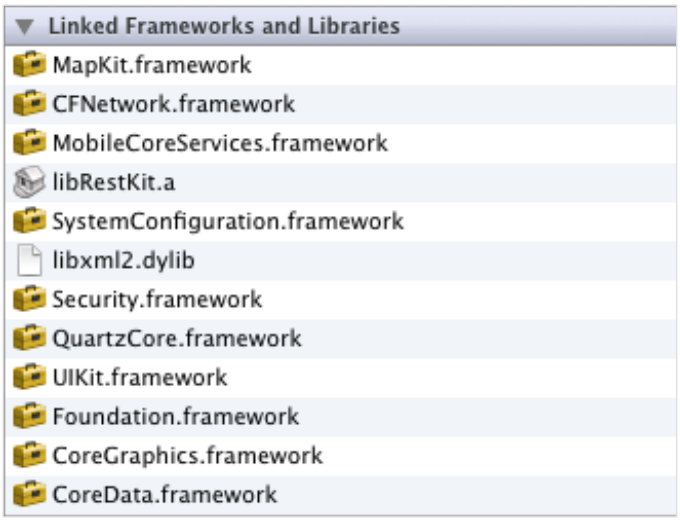


Figura 26 – Listado de frameworks utilizados en la aplicación

De entre todos los frameworks utilizados cabe destacar los siguientes:

- **RestKit**: es un conjunto de librerías de Objective-C para iOS que permite interactuar con servicios web de una manera simple y rápida. Combina una sencilla API de solicitud/respuesta de HTTP con un potente sistema de mapeo de objetos que ahorra una cantidad considerable de código. No viene incluido en Xcode, por lo que debe descargarse e implementarse en el proyecto manualmente. Se puede encontrar en: https://github.com/RestKit/RestKit

- **CoreData**: de todos los métodos de almacenamiento persistente de datos en iOS, Core Data es el mejor ya que reduce el gasto innecesario de memoria en las aplicaciones, aumenta la velocidad de respuesta, y evita tener que escribir una gran cantidad de código innecesario. Además, permite almacenar la información en un modelo relacional de entidad-atributo, lo que en este caso supone toda una ventaja.[10]

- **MapKit**: el conjunto Map Kit, ofrece una interfaz para incluir mapas directamente en las vistas de la aplicación. También permite realizar anotaciones en los mapas, añadir distintas capas, obtener las coordenadas de un punto en concreto, etc.[11]

- **QuartzCore**: Quartz Core permite procesar imágenes, insertar y manipular vídeos, o personalizar cualquier elemento gráfico de las aplicaciones. En este caso se ha utilizado para personalizar la tabla de Contacto dentro de las plantillas de Vicerrectorado, Centro y Departamento.[12]

4.4.2. Vistas de la aplicación (Storyboard)

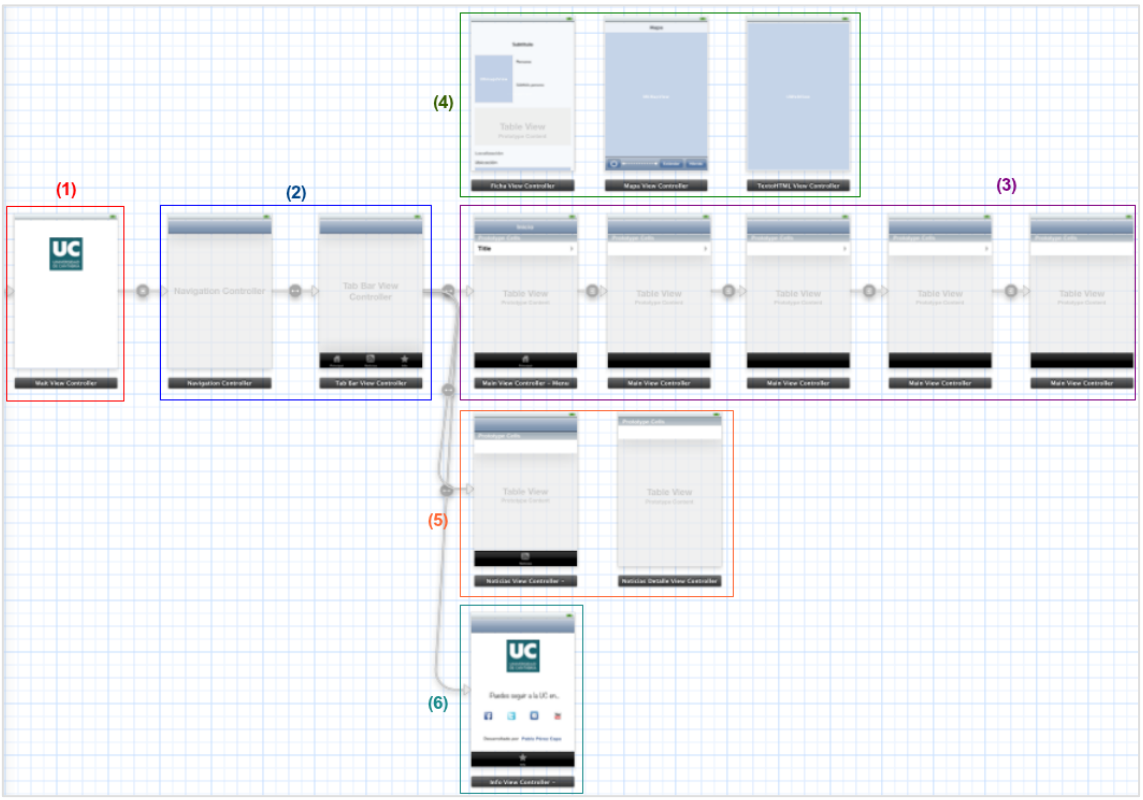


Figura 27 – Storyboard completo de la aplicación

La aplicación se compone de un total de 14 vistas, que pueden agruparse de la siguiente manera:

- **Vista de inicio (1)**: es la vista que se carga al arrancar la aplicación. En ella se muestra el resultado de las operaciones iniciales que comprueban la comunicación con el servidor y si hay alguna nueva versión del menú disponible.
- **Vistas de navegación (2)**: estas dos vistas permiten realizar la transición entre las distintas vistas mediante navegación y además dividen el contenido en tres secciones (*tabs*):
    - **Principal**: contiene información general de la UC estructurada mediante un menú.
    - **Noticias**: lector de feed RSS de noticias de la UC.
    - **Info**: contiene enlaces al perfil de la UC en diferentes redes sociales e información de contacto del desarrollador de la aplicación.

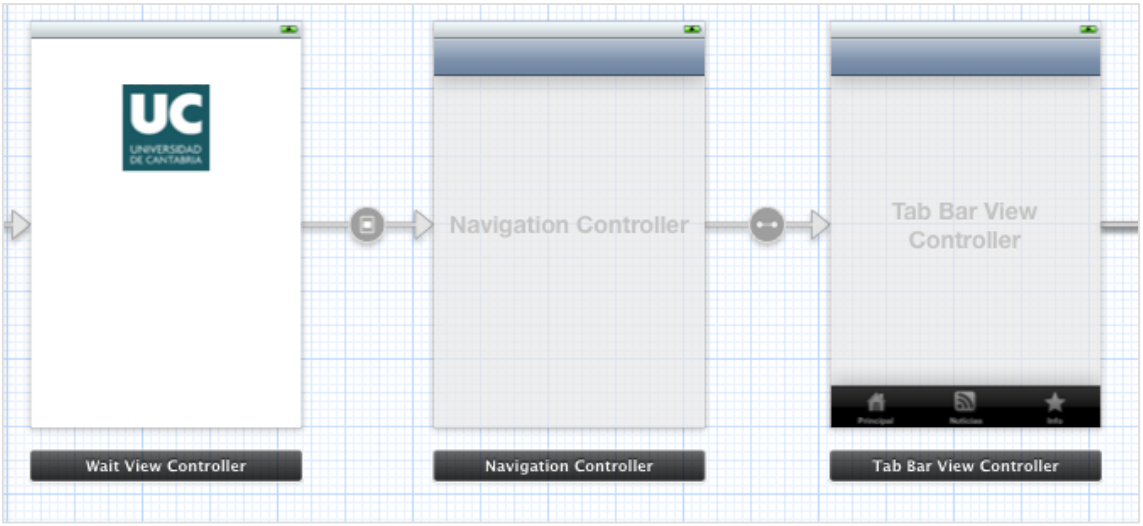


Figura 28 – Vista inicial y vistas de navegación

- **Vistas de menú (3)**: mostrarán los distintos niveles del menú de acceso a la información de la universidad.

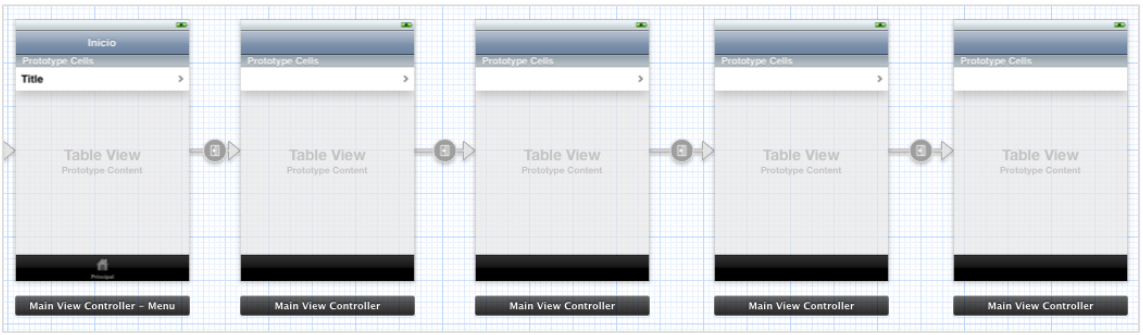


Figura 29 – Vistas para cargar el menú (hasta 5 niveles)

- **Vistas de página (4)**: vistas para cada tipo de página definida, que son:
  - **Ficha**: utilizada para mostrar información que incluya fundamentalmente una imagen, información de contacto y una ubicación mostrada en un mapa.
  - **Mapa**: esta vista se utiliza para mostrar distintas localizaciones en un mapa, además de la ubicación del usuario si este lo desea.
  - **TextoHTML**: muestra información que puede incluir texto enriquecido, imágenes y todo tipo de etiquetas HTML.

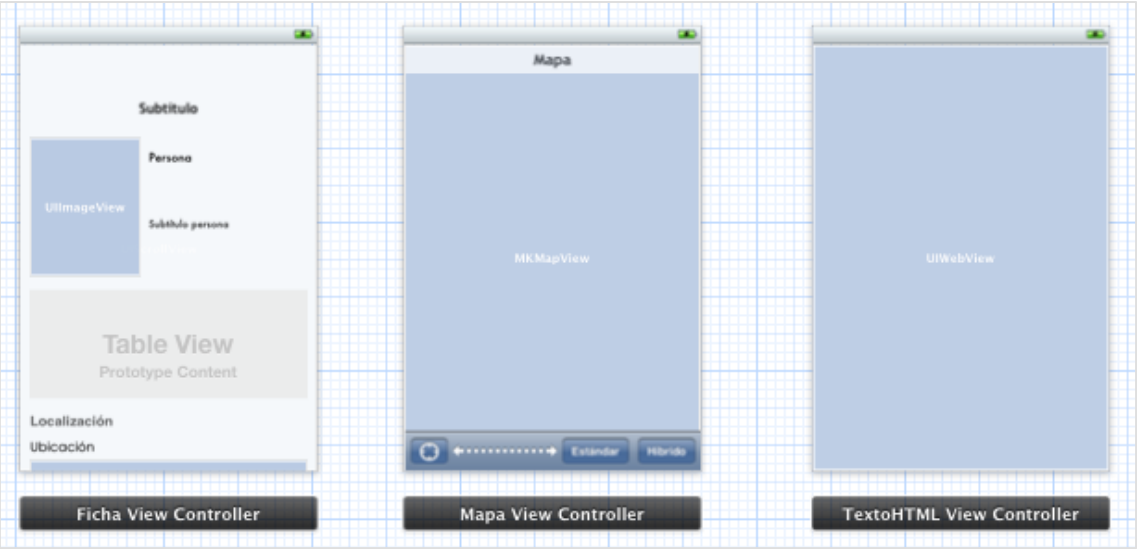


Figura 30 – Vistas para los distintos tipos de página

- **Vistas de noticias (5)**: la primera muestra un listado con las últimas noticias del feed del RSS de la UC y la segunda permitirá cargar el detalle de una noticia seleccionada, incluyendo el título, la hora de publicación, una descripción y un enlace para ver la noticia completa en el navegador web Safari.

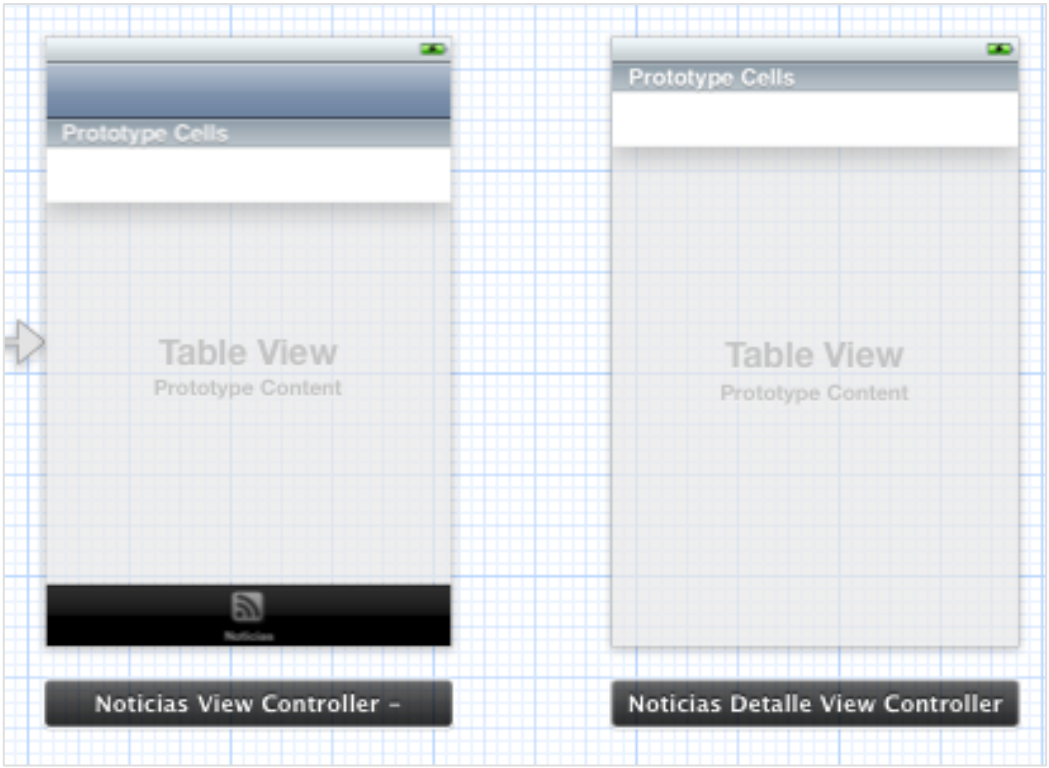


Figura 31 – Vistas para lectura de feed RSS de la UC

- **Vista de información (6)**: contiene una serie de enlaces a los perfiles de la UC en distintas redes sociales además de información de contacto del desarrollador de la aplicación.

Figura 32 – Vista de información de la UC

4.4.3. Controladores

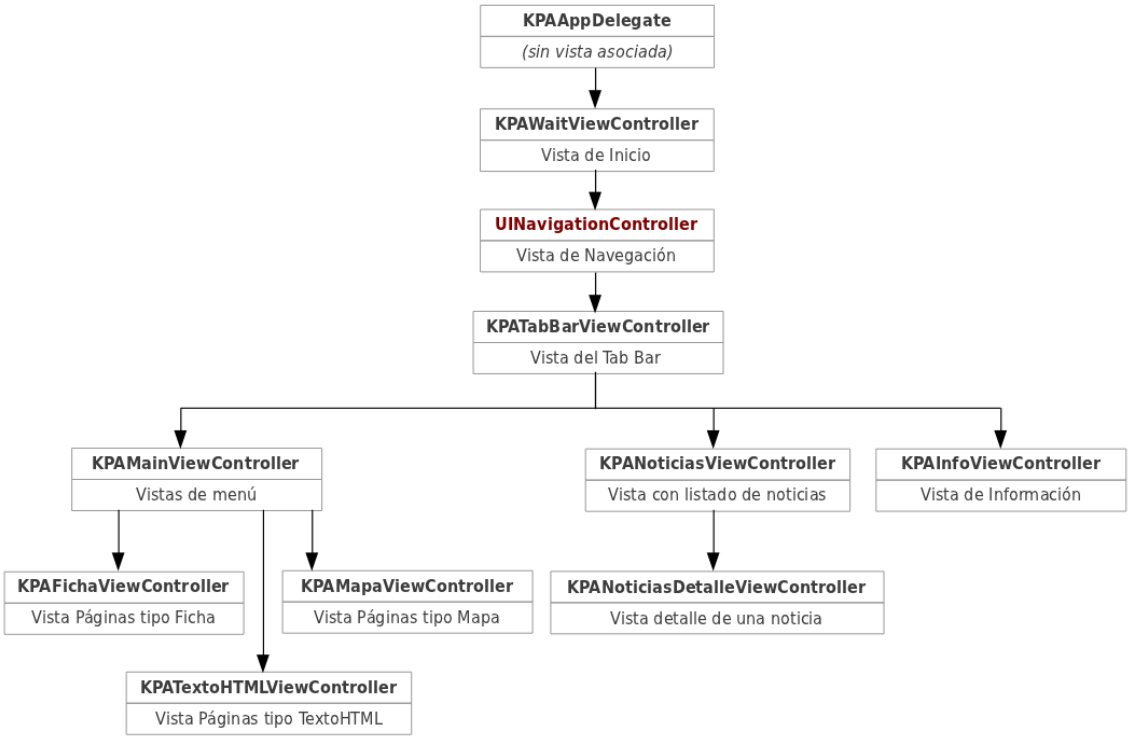


Figura 33 – Controladores utilizados en la aplicación y sus vistas asociadas

Estos son los controladores utilizados en la aplicación:

- **KPAAppDelegate**: delegado de la aplicación encargado de recibir las notificaciones cuando la aplicación alcanza ciertos estados (p. ej.: cuando entra

en *background*). Aquí se incluyen las operaciones que debe realizar la aplicación al inicio, que en este caso están relacionadas con el uso de *Core Data*, y la dirección IP del servidor Web, que podrá ser fácilmente accesible desde el resto de vistas.

- **KPAWaitViewController**: es el controlador de la vista de inicio. Aquí se realizan las operaciones iniciales cuando arranca la aplicación.

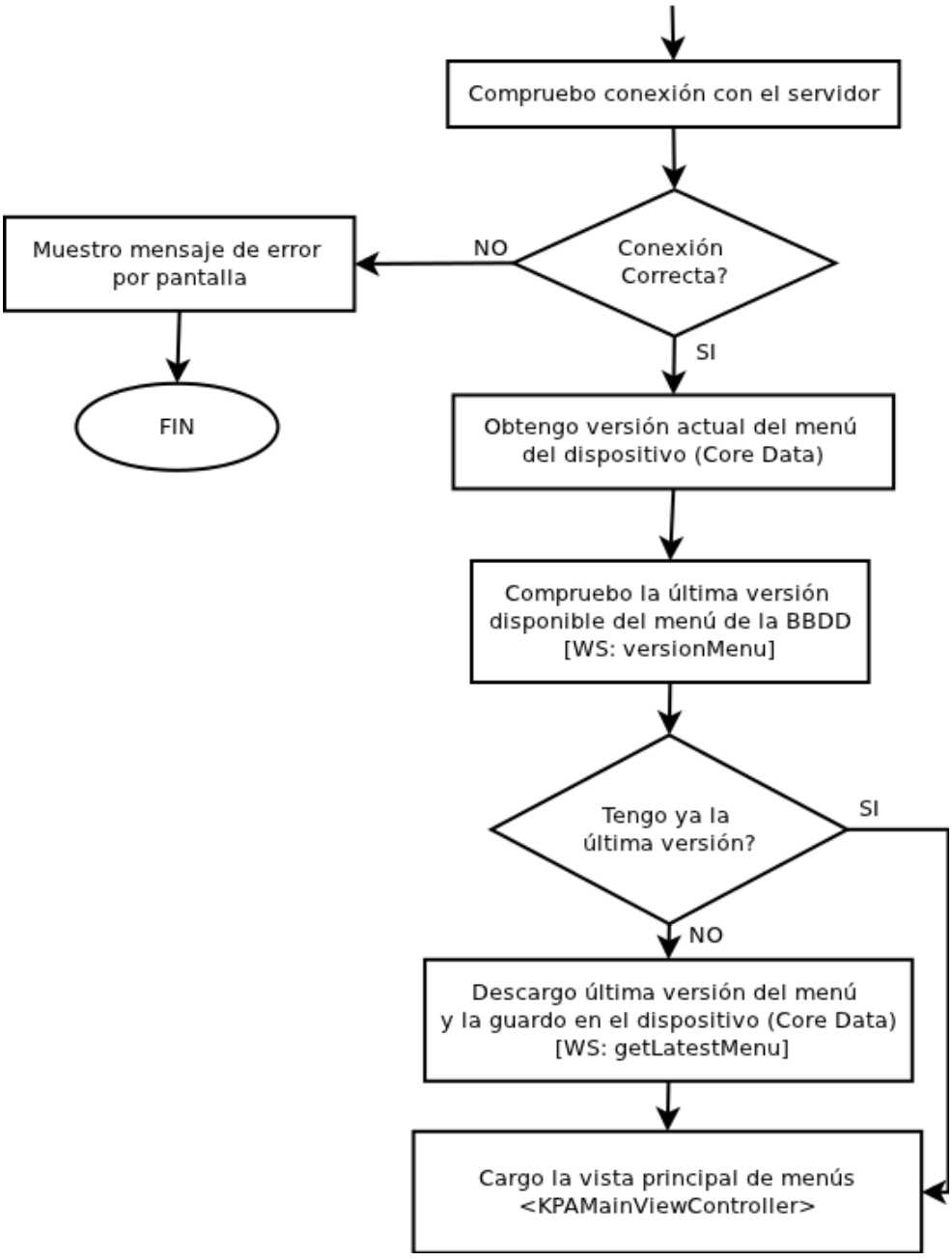


Figura 34 – Diagrama de flujo de KPAWaitController

- **UINavigationController**: controlador para el uso de navegación entre vistas. En este caso se ha utilizado el controlador que viene por defecto en Xcode.
- **KPATabBarViewController**: controlador para el uso de un tab bar, que como se ha comentado previamente, divide la aplicación en tres secciones: *Principal*,

Noticias e *Info*. En este caso se ha creado una clase personalizada a partir de la clase *UITabBarViewController* que viene por defecto en Xcode, que contiene una función que permite saber qué elemento del Tab Bar ha sido seleccionado y ponerle el título que le corresponda en cada caso.

- **KPAMainViewController**: controlador encargado de cargar el menú del dispositivo en las correspondientes vista de menú y de gestionar tanto la navegación entre las mismas como la carga de páginas (*Ficha, Mapa* o *TextoHTML*).

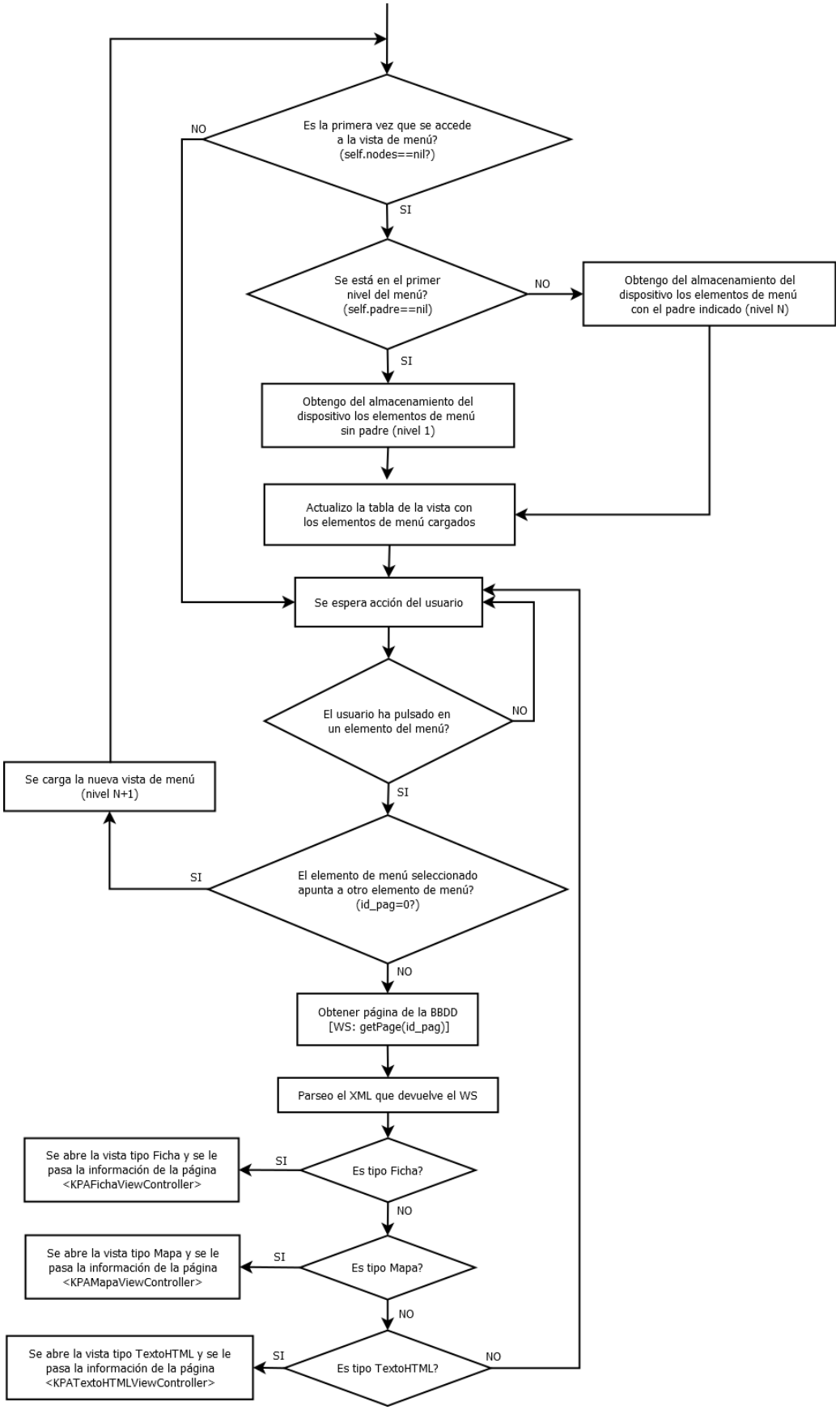


Figura 35 – Diagrama de flujo de KPAMainViewController

- **KPAFichaViewController**: carga la información de una página de tipo Ficha en una vista de tipo Ficha.
- **KPAMapaViewController**: carga una serie de localizaciones en un mapa. Además, incluye un botón que permite representar en dicho mapa la ubicación del usuario.
- **KPATextoHTMLViewController**: carga la información de una página de tipo TextoHTML en una vista de tipo TextoHTML.
- **KPANoticiasViewController**: controlador encargado de descargar el feed RSS de noticias de la UC y mostrarlas en una lista en la vista principal de noticias.

Figura 36 – Diagrama de flujo de KPANoticiasViewController

- **KPANoticiasDetalleViewController**: carga la información de una noticia (*RSSEntry*) en la vista de detalle de noticia.
- **KPAInfoViewController**: muestra la información de los perfiles de la UC en distintas redes sociales en la vista de información.

4.4.4. Modelo de Datos (Core Data)

Para agilizar la carga y uso de la aplicación se almacenará en un modelo de datos la estructura del menú utilizado en la sección Principal de la aplicación. Además se almacenará también el número de versión a la que corresponde dicho menú para poder comprobar si el servidor web dispone de alguna versión más actualizada y si es así descargarla.

El modelo de datos consta de dos entidades:

- **MenuItem**: almacena cada elemento del menú, que se corresponderá con una celda de la tabla de menú. Un elemento de menú podrá apuntar a otro elemento de menú o directamente a una página, que podrá ser de tipo Ficha, Mapa o TextoHTML.

  Un *MenuItem* contiene los siguientes atributos:

Atributo	Tipo	Descripción
id_menu	NSNumber	Identificador del elemento de menú
id_pag	NSNumber	Identificador de la página a la que apunta el elemento de menú, que será 0 si en vez de una página apunta a otro elemento de menú
padre	NSNumber	Elemento de menú del que nace el elemento actual
prioridad	NSNumber	Número que indica la posición en que debe situarse el elemento en la lista de elementos de menú mostrados (irán primero en la lista los de número de prioridad menor)
texto	NSString	Texto para mostrar en el menú (nombre del elemento)
Relaciones		
pertenece	Cada *MenuItem* pertenece a una sola Version (*el dispositivo sólo almacenará la versión más reciente del menú)	

- **Version**: para almacenar la versión actual del menú. Una entidad *Version* contiene el siguiente atributo:

Atributo	Tipo	Descripción
number	NSString	Número de la versión del menú almacenada en el dispositivo
Relaciones		
contiene	Una Version podrá contener varios *MenuItems*	

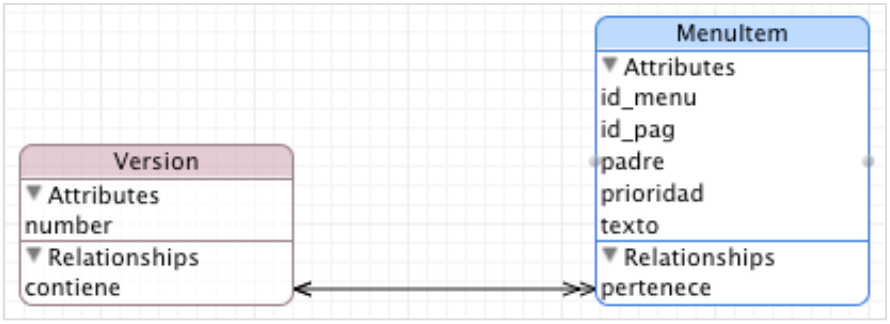


Figura 37 – Esquema del modelo de datos

Para facilitar el uso del modelo de datos se crearon tanto las clases correspondientes para cada entidad como un archivo para importarlas todas de una sola vez (*modelo.h*).

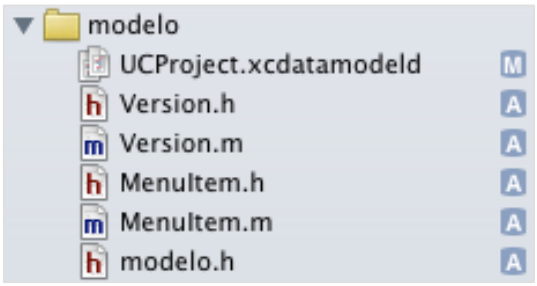


Figura 38 – Listado de archivos correspondientes al modelo de datos

4.4.5. Acceso a los servicios Web

Para acceder a los servicios web se utiliza el delegado *RKRequestDelegate* de la librería RestKit. Este delegado incluye una serie de funciones que simplifican la comunicación con los servicios Web.

Para hacer una petición a un servicio Web el proceso es el siguiente:

- Iniciar el cliente RestKit (RKClient) con la dirección del servidor web
- Hacer una petición HTTP GET a la función del servicio web que se desee pasándole los parámetros como un diccionario si los requiere
- Procesar la respuesta en el método del delegado *didLoadResponse*.

Como ejemplo, para obtener la última versión del menú estas son las funciones utilizadas, que serán llamadas en este orden: initClient → getLatestMenu → didLoadResponse

```
////3. WEB SERVICES
-(void)initClient{
    self.client = [RKClient clientWithBaseURLString:[NSString
stringWithFormat:@"http://%@/proyecto/",self.serverURL]];
}

-(void)getLatestMenu{
    self.reqOperation=@"getLatestMenu";
    self.infoLabel.text=@"Descargando nueva versión del menú";
    [self.client get:@"/WebService1.asmx/getLatestMenu" delegate:self];
}

#pragma mark RKRequestDelegate methods
- (void)request:(RKRequest*)request didLoadResponse:(RKResponse*)response {
    if ([request isGET]) {
        [self parseMenuData:[response bodyAsString]];
    }else if ([response isNotFound])
    {       // No ha devuelto nada, error
        [self showAlert:[NSString stringWithFormat:@"El recurso con ruta '%@'
no ha sido encontrado.", [request resourcePath]]];
    }else{
        [self showAlert:[response bodyAsString]];
    }
}
```

Figura 39 – Obtención de la última versión del menú (KPAWaitViewController)

4.4.6. Lectura de XML

Para la lectura de los datos en formato XML recibidos tanto por los servicios web como por el feed RSS se utiliza el delegado NSXMLParser.

El proceso para la lectura de datos XML también es muy sencillo:

- Se crea el parser XML y se le pasan los datos en formato NSData (función: *parseMenuData*)
- Si hay algún error durante el parseo se mostrará una traza por consola (sólo modo Debug, función: *parseErrorOccurred*)
- Cada vez que se encuentre un tag de inicio de elemento (p.ej.: <titulo>) se lanzará la función *didStartElement*. Ahí simplemente se reiniciará el valor de la variable que almacenará el contenido de cada nodo (*self.nodevalue*).
- Cada vez que se encuentren caracteres contenidos entre un tag de inicio y otro de fin se llama a la función *foundCharacters*. En esta función se almacenan todos los caracteres encontrados en la variable que contiene el valor de cada nodo (*self.nodevalue*).
- Cada vez que se encuentre un tag de fin de elemento (p.ej.: </titulo>) se lanzará la función *didEndElement*. Aquí lo que se hace es extraer el valor acumulado para cada nodo (*self.nodevalue*) y se almacenan, por ejemplo, en un diccionario que asociará cada tag (*elementName*) con su valor.
- Por último, al terminar de parsear el documento se llama a la función *parserDidEndDocument* desde la que se comprobarán los resultados y se realizarán las operaciones convenientes en cada caso.

En el siguiente ejemplo se pueden ver las funciones utilizadas para parsear los XML recibidos en KPAWaitViewController, que en este caso procesa datos que pueden venir de dos operaciones diferentes: *getLatestMenu* o *checkMenuVersion*.

```
////4. XMLREADER
-(void)parseMenuData:(NSString *)menuData{
    NSData *data=[menuData dataUsingEncoding:NSUTF8StringEncoding];
    NSXMLParser *parser = [[NSXMLParser alloc] initWithData:data];

    [parser setDelegate:self];
    [parser parse];
}

-(void)parser:(NSXMLParser *)parser parseErrorOccurred:(NSError *)parseError{
    NSLog(@"Parser error:%@",parseError);
}

-(void)parser:(NSXMLParser *)parser didStartElement:(NSString *)elementName
namespaceURI:(NSString *)namespaceURI qualifiedName:(NSString *)qName
attributes:(NSDictionary *)attributeDict{
    self.nodeValue=@"";
}

-(void)parser:(NSXMLParser *)parser didEndElement:(NSString *)elementName
namespaceURI:(NSString *)namespaceURI qualifiedName:(NSString *)qName{
    if([self.reqOperation isEqualToString:@"getLatestMenu"]){
        if ([elementName isEqualToString:@"menu"]) {
            NSDictionary *tmp=[[NSDictionary alloc]
initWithDictionary:self.tmpMenuNode];
            [self.menus addObject:tmp];
        }else if ([validNodes containsObject:elementName]) {
            [self.tmpMenuNode setValue:self.nodeValue forKey:elementName];
        }
    }else if([self.reqOperation isEqualToString:@"checkMenuVersion"]){
        self.latestVersion=self.nodeValue;
    }
    self.nodeValue=@"";
}
-(void)parserDidEndDocument:(NSXMLParser *)parser{
    if ([self.reqOperation isEqualToString:@"checkMenuVersion"]){
        if ([self.latestVersion isEqualToString:self.currentVersion]) {
            //TRUE: Cargar menu de CoreData
            [self loadMenu];
        }else {
            //FALSE: descargar ultima version
            [self getLatestMenu];
        }
    }else if([self.reqOperation isEqualToString:@"getLatestMenu"]){
        [self saveMenuItems:self.menus];
        [self loadMenu];
    }
}

-(void)parser:(NSXMLParser *)parser foundCharacters:(NSString *)string{
    self.nodeValue=[NSString stringWithFormat:@"%@%@",self.nodeValue,string];
}
```

Figura 40 – Ejemplo de parseo de datos XML (KPAWaitViewController)

5. PUESTA EN MARCHA

Para poner en funcionamiento el proyecto, he tenido que utilizar varias herramientas adicionales que detallaré a continuación.

5.1. INTERNET INFORMATION SERVICES 7 (IIS7)

Internet Information Services, de Microsoft, es un servidor web y un conjunto de herramientas y funciones necesarias para administrarlo.

Figura 41 – Pantalla de bienvenida de IIS7

En este caso, el IIS se ha utilizado para alojar los servicios web y por tanto, que pudieran ser accedidos de forma remota tanto en red local como desde Internet.

Para ello, he tenido que seleccionar un puerto del router que no estuviera en uso, que en este caso ha sido el 8070.[13]

5.2. NO-IP DYNAMIC UPDATE CLIENT

Debido a que el servidor en este caso es mi ordenador portátil, dependiendo de la localización desde la que me conectase, tenía una IP pública u otra, lo que me impedía dar una ruta fija a la aplicación para que accediera a los servicios web. Gracias a esta herramienta, que automáticamente detecta esa IP pública y la enmascara tras una dirección fija (http://kpa.hopto.org, en este caso), consigo subsanar este problema, lo que unido a la configuración anterior del IIS7 establece como dirección fija de mis web services: http://kpa.hopto.org:8070/proyecto/WebService1.asmx

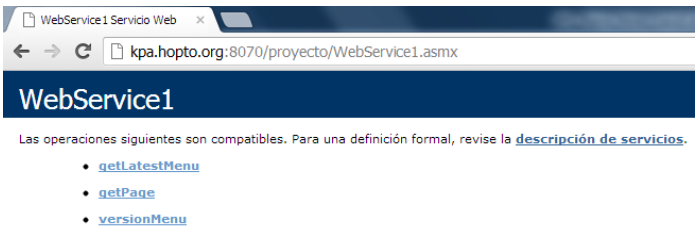


Figura 42 – Acceso al servicio web con IP fija

6. RESULTADOS OBTENIDOS

Tras realizar los ajustes mencionados, la aplicación ya pasa a ser totalmente operativa, con un funcionamiento que se detalla a continuación.

Inicio de la aplicación

Pulsando el icono correspondiente, la aplicación arranca y da paso a la vista de espera.

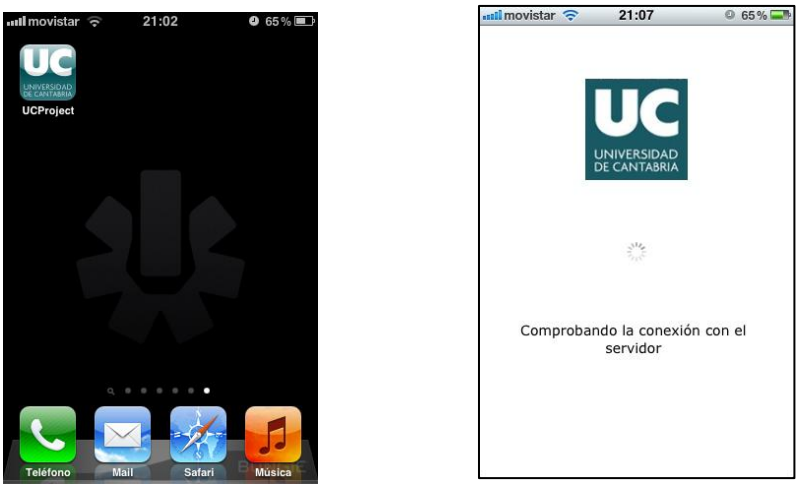


Figura 43 – Icono en el dispositivo y vista de espera

En este punto, si la conexión con el servidor es satisfactoria procederá a la comprobación de la versión su menú respecto al de la base de datos y, en caso de que sea necesario, lo actualizará. Por otro lado, si la conexión falla, aparecerá un mensaje de error.

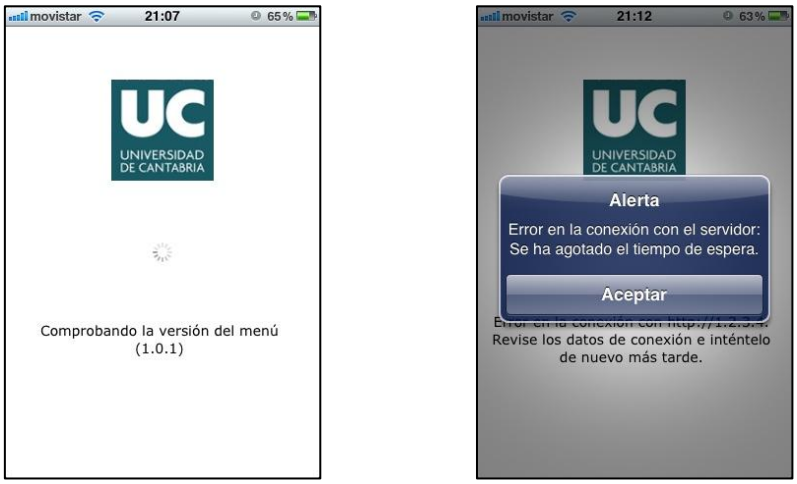


Figura 44 – Comprobación de menú y error

Ya con la versión más actual del menú, la aplicación arranca mostrando los elementos superiores del mismo, así como los tres botones del tab bar (barra inferior).

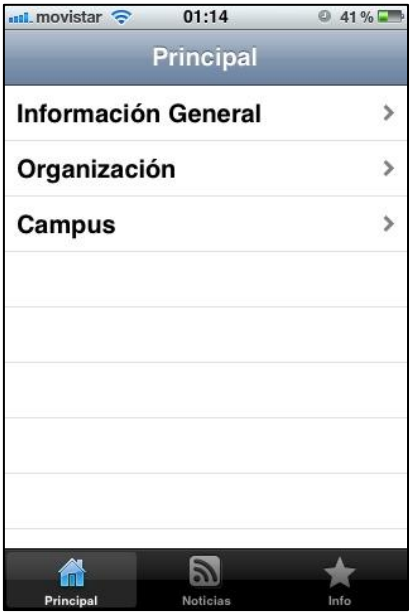


Figura 45 – Menú principal

Principal

Centrándonos por ahora en la sección Principal, si seguimos navegando, cada uno de los elementos actuales da paso a nuevas opciones.

Figura 46 – Menús de Información, Organización y Campus

En este nivel, la sección Campus ya ha llegado al final de sus vistas de menú, por lo que al pulsar en cualquiera de sus opciones, cargará la página correspondiente:

Figura 47 – Campus de Los Castros y Torrelavega

Lo mismo ocurre con la sección Información General:

Figura 48 – Ejemplo de página de Información General

Sin embargo, en la sección Organización todavía resta un nivel de navegación.

Figura 49 – Menús de Vicerrectorados, Centros y Departamentos

Ahora sí, al pulsar sobre cualquiera de estas opciones, cargará la ficha correspondiente.

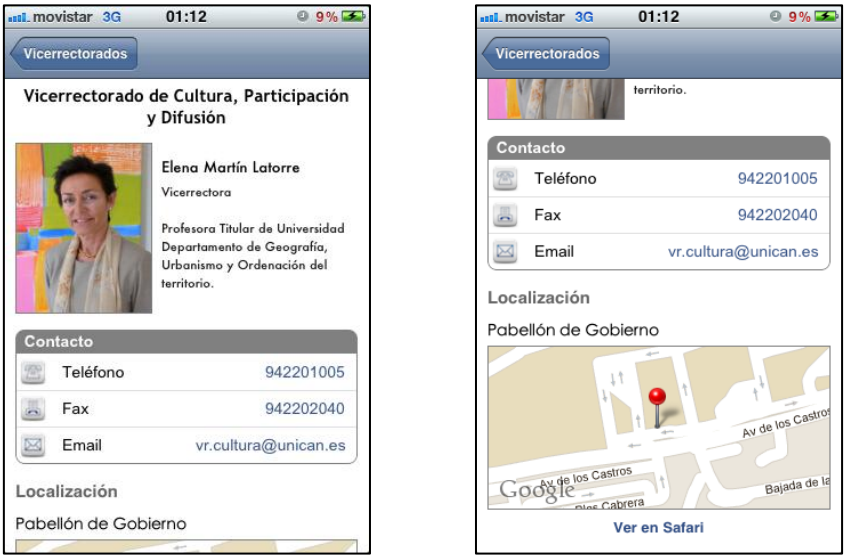


Figura 50 – Ejemplo de ficha de Vicerrectorado (a la derecha scroll-down)

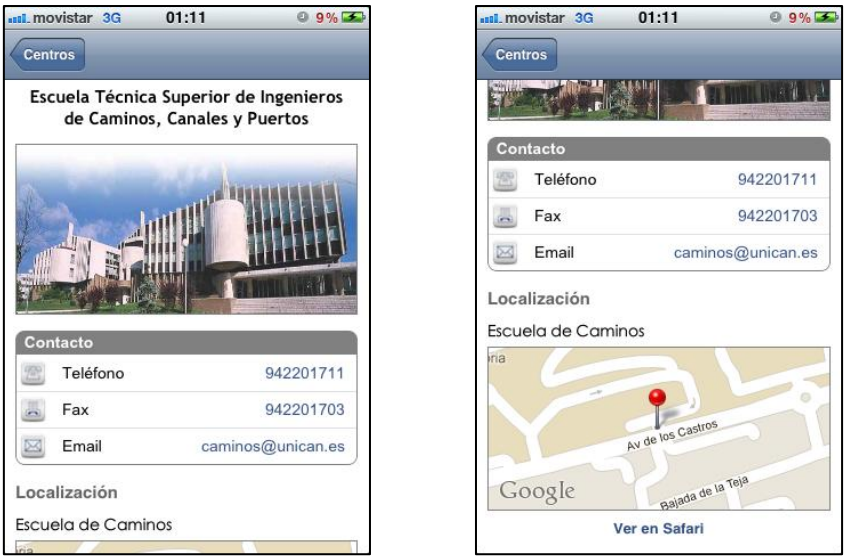


Figura 51 – Ejemplo de ficha de Centro (a la derecha scroll-down)

Figura 52 – Ejemplo de ficha de Departamento (a la derecha scroll-down)

Hasta aquí llega la navegación de la sección principal.

Noticias

A continuación se encuentra la sección de noticias, que carga el feed directamente del servidor de la Universidad, estando por lo tanto totalmente actualizado en todo momento.

Simplemente hay que pulsar el botón correspondiente en el tab bar y se mostrarán las noticias más actuales:

Figura 53 – Ejemplo de feed de Noticias

Ahora, sí se selecciona una de ellas aparecerá una nueva vista con una descripción de la misma y el enlace para ver la noticia completa en Safari.

Figura 54 – Detalle de una noticia

Info

Por último está la página Info, una sencilla vista con enlaces los perfiles de la UC en varias redes sociales, así como un enlace con un email de contacto.

Figura 55 – Vista Info

7. CONCLUSIÓN

Tras la realización de este proyecto, he llegado a varias conclusiones, tanto en cuanto a la metodología como a las herramientas y tecnologías que he utilizado.

En primer lugar debo destacar la importancia de dedicar un tiempo considerable a la planificación del proyecto. En vez comenzar programando la base de datos e ir resolviendo los problemas sobre la marcha, dediqué bastante tiempo al diseño de las tablas y de las plantillas que iban a servir como cimientos del proyecto. De este modo, apenas he tenido que volver sobre mis pasos, lo que terminó por ahorrarme una importante cantidad de tiempo.

En cuanto a las tecnologías empleadas, la utilización de una arquitectura de 3-capas en el proyecto me ha servido para utilizar herramientas y entornos de desarrollo que desconocía por completo. Con SQL Server he podido crear desde cero y después gestionar una base de datos relacional con cierto nivel de complejidad, que luego ha podido ser accedida desde Visual Studio debido a la perfecta integración entre ambos.

Quizá lo que mayor incertidumbre podía crear en un principio era cómo se iban a relacionar estos entornos de Microsoft con el Xcode de Apple. En general gracias a la elección de un formato bastante extendido como el XML y al Restkit Framework la transición fue lo suficientemente fluida, por lo que creo que el uso de esta arquitectura para este tipo de proyecto ha sido todo un acierto.

En definitiva, se ha conseguido crear un CMS con un alto nivel de escalabilidad, y que sienta las bases para poder incrementar tanto el número de servicios que ofrece, como el de dispositivos soportados. Además, la implementación para iPhone ha dado como resultado una forma intuitiva y cómoda de acceder a toda esa información con una aplicación totalmente funcional.

8. POSIBLES MEJORAS

Uno de los más claros objetivos de cara al futuro de este proyecto sería su implementación en nuevas plataformas (Android, Windows Phone, etc…), que como ya se ha comentado, se lograría con tan solo desarrollar la capa de presentación correspondiente. Personalmente considero que la implementación más necesaria sería la de una aplicación de escritorio para Windows (dado que es el sistema operativo que se usa mayoritariamente en la UC) que permitiera gestionar los contenidos de la manera más intuitiva posible, ya que actualmente los datos han de ser introducidos en la base de datos de manera manual.

Además, hay varias limitaciones que vienen dadas porque el sistema no está implementado en los servidores de la Universidad, por lo que de implementarse, convendría adaptar la capa de acceso a datos para que pudiera consumir información de otros orígenes, como las bases de datos propias de la Universidad, lo que permitiría también incluir nuevos servicios como el Campus Virtual o la consulta de correo electrónico. También se podrían incluir más canales de noticias o incluso la radio del SIDE, en un intento por aunar el máximo de prestaciones en la aplicación y darle así un valor añadido sobre el acceso a la página web desde el navegador.

Por último, si los servicios web se hacen públicos, habría que hacer un estudio sobre su vulnerabilidad y tratar de incrementar su seguridad.

9. REFERENCIAS

[1]	B. Herrero Hurtado. *Desarrollo Web con Drupal*. Universidad Rey Juan Carlos, Curso 2009-2010.
[2]	*Extensible Markup Language*. Wikipedia. http://es.wikipedia.org/wiki/Extensible_Markup_Language
[3]	*Tecnologías SQL Server 2008*. Microsoft Corporation. http://www.microsoft.com/latam/sql/2008/technologies/default.mspx
[4]	*Use SQL Management Studio*. Microsoft MSDN. http://msdn.microsoft.com/en-us/library/ms174173.aspx
[5]	*Microsoft Visual Studio*. Wikipedia. http://es.wikipedia.org/wiki/Microsoft_Visual_Studio
[6]	*Resumen de las características de C#*. Microsoft MSDN. http://msdn.microsoft.com/es-es/library/aa287483(v=vs.71).aspx
[7]	D. Arias Vazquez. *¿Qué es Objective-C?*. Universidad de A Coruña. http://sabia.tic.udc.es/gc/Contenidos%20adicionales/trabajos/Interfaces/mac/cocoa3_1.html
[8]	*Restricciones PRIMARY KEY*. Microsoft MSDN. http://msdn.microsoft.com/es-es/library/ms191236(v=sql.105).aspx
[9]	*Clave foránea*. Wikipedia. http://es.wikipedia.org/wiki/Clave_for%C3%A1nea
[10]	*Core Data on iOS5 Tutorial*. RayWenderlich.com. 13/04/2010. http://www.raywenderlich.com/934/core-data-on-ios-5-tutorial-getting-started
[11]	*Map Kit Framework Reference*. iOS Developer Library. 19/09/2012. http://developer.apple.com/library/ios/#documentation/MapKit/Reference/MapKit_Framework_Reference/_index.html
[12]	*Quartz Core Framework Reference*. Mac Developer Library. 09-01-2012. http://developer.apple.com/library/mac/#documentation/graphicsimaging/reference/QuartzCoreRefCollection/_index.html
[13]	*Internet Information Services*. Wikipedia. http://es.wikipedia.org/wiki/Internet_Information_Services

Apéndice I. Script BBDD

```
USE [master]
GO
/****** Object:  Database [UC]    Script Date: 10/15/2012 02:08:45 ******/
CREATE DATABASE [UC] ON  PRIMARY
( NAME = N'UC', FILENAME = N'C:\Program Files\Microsoft SQL
Server\MSSQL10_50.MSSQLSERVER\MSSQL\DATA\UC.mdf' , SIZE = 3072KB , MAXSIZE =
UNLIMITED, FILEGROWTH = 1024KB )
 LOG ON
( NAME = N'UC_log', FILENAME = N'C:\Program Files\Microsoft SQL
Server\MSSQL10_50.MSSQLSERVER\MSSQL\DATA\UC_log.ldf' , SIZE = 1024KB , MAXSIZE
= 2048GB , FILEGROWTH = 10%)
GO
ALTER DATABASE [UC] SET COMPATIBILITY_LEVEL = 100
GO
IF (1 = FULLTEXTSERVICEPROPERTY('IsFullTextInstalled'))
begin
EXEC [UC].[dbo].[sp_fulltext_database] @action = 'enable'
end
GO
ALTER DATABASE [UC] SET ANSI_NULL_DEFAULT OFF
GO
ALTER DATABASE [UC] SET ANSI_NULLS OFF
GO
ALTER DATABASE [UC] SET ANSI_PADDING OFF
GO
ALTER DATABASE [UC] SET ANSI_WARNINGS OFF
GO
ALTER DATABASE [UC] SET ARITHABORT OFF
GO
ALTER DATABASE [UC] SET AUTO_CLOSE ON
GO
ALTER DATABASE [UC] SET AUTO_CREATE_STATISTICS ON
GO
ALTER DATABASE [UC] SET AUTO_SHRINK OFF
GO
ALTER DATABASE [UC] SET AUTO_UPDATE_STATISTICS ON
GO
ALTER DATABASE [UC] SET CURSOR_CLOSE_ON_COMMIT OFF
GO
ALTER DATABASE [UC] SET CURSOR_DEFAULT  GLOBAL
GO
ALTER DATABASE [UC] SET CONCAT_NULL_YIELDS_NULL OFF
GO
ALTER DATABASE [UC] SET NUMERIC_ROUNDABORT OFF
GO
ALTER DATABASE [UC] SET QUOTED_IDENTIFIER OFF
GO
ALTER DATABASE [UC] SET RECURSIVE_TRIGGERS OFF
GO
ALTER DATABASE [UC] SET  DISABLE_BROKER
GO
ALTER DATABASE [UC] SET AUTO_UPDATE_STATISTICS_ASYNC OFF
GO
ALTER DATABASE [UC] SET DATE_CORRELATION_OPTIMIZATION OFF
GO
ALTER DATABASE [UC] SET TRUSTWORTHY OFF
GO
ALTER DATABASE [UC] SET ALLOW_SNAPSHOT_ISOLATION OFF
GO
ALTER DATABASE [UC] SET PARAMETERIZATION SIMPLE
GO
```

```
ALTER DATABASE [UC] SET READ_COMMITTED_SNAPSHOT OFF
GO
ALTER DATABASE [UC] SET HONOR_BROKER_PRIORITY OFF
GO
ALTER DATABASE [UC] SET  READ_WRITE
GO
ALTER DATABASE [UC] SET RECOVERY SIMPLE
GO
ALTER DATABASE [UC] SET  MULTI_USER
GO
ALTER DATABASE [UC] SET PAGE_VERIFY CHECKSUM
GO
ALTER DATABASE [UC] SET DB_CHAINING OFF
GO
USE [UC]
GO
/****** Object:  User [IIS APPPOOL\DefaultAppPool]    Script Date: 10/15/2012
02:08:45 ******/
CREATE USER [IIS APPPOOL\DefaultAppPool] FOR LOGIN [IIS
APPPOOL\DefaultAppPool]
GO
/****** Object:  Role [WebAppUserRole]    Script Date: 10/15/2012 02:08:45
******/
CREATE ROLE [WebAppUserRole] AUTHORIZATION [dbo]
GO
/****** Object:  Schema [WebAppUserRole]    Script Date: 10/15/2012 02:08:46
******/
CREATE SCHEMA [WebAppUserRole] AUTHORIZATION [WebAppUserRole]
GO
/****** Object:  Schema [IIS APPPOOL\DefaultAppPool]    Script Date:
10/15/2012 02:08:46 ******/
CREATE SCHEMA [IIS APPPOOL\DefaultAppPool] AUTHORIZATION [IIS
APPPOOL\DefaultAppPool]
GO
/****** Object:  Table [dbo].[Plantillas]    Script Date: 10/15/2012 02:08:52
******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[Plantillas](
       [ID_PLANTILLA] [int] IDENTITY(1,1) NOT NULL,
       [NOMBRE] [varchar](30) NULL,
 CONSTRAINT [PK_Plantillas] PRIMARY KEY CLUSTERED
(
       [ID_PLANTILLA] ASC
)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS  = ON, ALLOW_PAGE_LOCKS  = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
SET ANSI_PADDING OFF
GO
/****** Object:  Table [dbo].[Paginas]    Script Date: 10/15/2012 02:08:52
******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[Paginas](
       [ID_PAG] [int] IDENTITY(1,1) NOT NULL,
       [NOMBRE] [varchar](60) NULL,
       [ID_PLANTILLA] [int] NOT NULL,
 CONSTRAINT [PK__Paginas__20AF02C0145C0A3F] PRIMARY KEY CLUSTERED
(
```

```
	[ID_PAG] ASC
)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS  = ON, ALLOW_PAGE_LOCKS  = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
SET ANSI_PADDING OFF
GO
/****** Object:  Table [dbo].[Elementos]    Script Date: 10/15/2012 02:08:52
******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[Elementos](
	[ELEM_PLANTILLA] [int] IDENTITY(1,1) NOT NULL,
	[NOMBRE] [varchar](30) NOT NULL,
	[TIPO] [varchar](20) NOT NULL,
	[ID_PLANTILLA] [int] NOT NULL,
	[CONTENIDO] [tinyint] NOT NULL,
	[MULTIPLE] [tinyint] NOT NULL,
	[PADRE] [int] NULL,
	[ORDEN] [int] NULL,
	[ETIQUETA] [varchar](30) NULL,
	[OPCIONAL] [tinyint] NOT NULL,
 CONSTRAINT [PK__Elemento__FF17647803317E3D] PRIMARY KEY CLUSTERED
(
	[ELEM_PLANTILLA] ASC
)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS  = ON, ALLOW_PAGE_LOCKS  = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
SET ANSI_PADDING OFF
GO
/****** Object:  Table [dbo].[Menu]    Script Date: 10/15/2012 02:08:52
******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[Menu](
	[ID_MENU] [int] IDENTITY(1,1) NOT NULL,
	[Texto] [varchar](max) NOT NULL,
	[Padre] [int] NULL,
	[Prioridad] [int] NULL,
	[ID_PAG] [int] NULL,
	[Vers] [varchar](12) NOT NULL,
 CONSTRAINT [ID_MENU_PK_MENU] PRIMARY KEY CLUSTERED
(
	[ID_MENU] ASC
)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS  = ON, ALLOW_PAGE_LOCKS  = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
SET ANSI_PADDING OFF
GO
/****** Object:  Table [dbo].[Datos]    Script Date: 10/15/2012 02:08:52
******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[Datos](
```

```
	[ID_DATO] [int] IDENTITY(1,1) NOT NULL,
	[ELEM_PLANTILLA] [int] NOT NULL,
	[VALOR] [varchar](max) NULL,
PRIMARY KEY CLUSTERED
(
	[ID_DATO] ASC
)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS  = ON, ALLOW_PAGE_LOCKS  = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
SET ANSI_PADDING OFF
GO
/****** Object:  Table [dbo].[Ordenacion]    Script Date: 10/15/2012 02:08:52
******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Ordenacion](
	[ID_PAG] [int] NOT NULL,
	[ELEM_PLANTILLA] [int] NOT NULL,
	[ID_DATO] [int] NOT NULL,
	[ORDEN] [int] NULL,
	[ID_ORDENACION] [int] IDENTITY(1,1) NOT NULL,
 CONSTRAINT [ID_ORDENACION_PK_ORDENACION] PRIMARY KEY CLUSTERED
(
	[ID_ORDENACION] ASC
)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS  = ON, ALLOW_PAGE_LOCKS  = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
/****** Object:  Default [DF__Elementos__CONTE__0519C6AF]    Script Date:
10/15/2012 02:08:52 ******/
ALTER TABLE [dbo].[Elementos] ADD  CONSTRAINT [DF__Elementos__CONTE__0519C6AF]
DEFAULT ((1)) FOR [CONTENIDO]
GO
/****** Object:  Default [DF__Elementos__MULTI__060DEAE8]    Script Date:
10/15/2012 02:08:52 ******/
ALTER TABLE [dbo].[Elementos] ADD  CONSTRAINT [DF__Elementos__MULTI__060DEAE8]
DEFAULT ((0)) FOR [MULTIPLE]
GO
/****** Object:  Default [DF_Elementos_OPCIONAL]    Script Date: 10/15/2012
02:08:52 ******/
ALTER TABLE [dbo].[Elementos] ADD  CONSTRAINT [DF_Elementos_OPCIONAL]  DEFAULT
((0)) FOR [OPCIONAL]
GO
/****** Object:  Default [DF__Menu__Vers__32E0915F]    Script Date: 10/15/2012
02:08:52 ******/
ALTER TABLE [dbo].[Menu] ADD  DEFAULT ('1.0.0') FOR [Vers]
GO
/****** Object:  ForeignKey [PLANTILLAS_ID_PLANTILLA_FK_PAGINAS]    Script
Date: 10/15/2012 02:08:52 ******/
ALTER TABLE [dbo].[Paginas]  WITH CHECK ADD  CONSTRAINT
[PLANTILLAS_ID_PLANTILLA_FK_PAGINAS] FOREIGN KEY([ID_PLANTILLA])
REFERENCES [dbo].[Plantillas] ([ID_PLANTILLA])
GO
ALTER TABLE [dbo].[Paginas] CHECK CONSTRAINT
[PLANTILLAS_ID_PLANTILLA_FK_PAGINAS]
GO
/****** Object:  ForeignKey [FK_Elementos_PadreHijo]    Script Date:
10/15/2012 02:08:52 ******/
ALTER TABLE [dbo].[Elementos]  WITH CHECK ADD  CONSTRAINT
[FK_Elementos_PadreHijo] FOREIGN KEY([PADRE])
REFERENCES [dbo].[Elementos] ([ELEM_PLANTILLA])
GO
ALTER TABLE [dbo].[Elementos] CHECK CONSTRAINT [FK_Elementos_PadreHijo]
GO
/****** Object:  ForeignKey [PLANTILLAS_ID_PLANTILLA_FK]    Script Date:
```

```
10/15/2012 02:08:52 ******/
ALTER TABLE [dbo].[Elementos]  WITH CHECK ADD  CONSTRAINT
[PLANTILLAS_ID_PLANTILLA_FK] FOREIGN KEY([ID_PLANTILLA])
REFERENCES [dbo].[Plantillas] ([ID_PLANTILLA])
GO
ALTER TABLE [dbo].[Elementos] CHECK CONSTRAINT [PLANTILLAS_ID_PLANTILLA_FK]
GO
/****** Object:  ForeignKey [FK_Menu_Paginas]    Script Date: 10/15/2012
02:08:52 ******/
ALTER TABLE [dbo].[Menu]  WITH CHECK ADD  CONSTRAINT [FK_Menu_Paginas] FOREIGN
KEY([ID_PAG])
REFERENCES [dbo].[Paginas] ([ID_PAG])
GO
ALTER TABLE [dbo].[Menu] CHECK CONSTRAINT [FK_Menu_Paginas]
GO
/****** Object:  ForeignKey [PADRE_FK_ID_MENU]    Script Date: 10/15/2012
02:08:52 ******/
ALTER TABLE [dbo].[Menu]  WITH CHECK ADD  CONSTRAINT [PADRE_FK_ID_MENU]
FOREIGN KEY([Padre])
REFERENCES [dbo].[Menu] ([ID_MENU])
GO
ALTER TABLE [dbo].[Menu] CHECK CONSTRAINT [PADRE_FK_ID_MENU]
GO
/****** Object:  ForeignKey [ELEMENTOS_ELEM_PLANTILLA_FK]    Script Date:
10/15/2012 02:08:52 ******/
ALTER TABLE [dbo].[Datos]  WITH CHECK ADD  CONSTRAINT
[ELEMENTOS_ELEM_PLANTILLA_FK] FOREIGN KEY([ELEM_PLANTILLA])
REFERENCES [dbo].[Elementos] ([ELEM_PLANTILLA])
GO
ALTER TABLE [dbo].[Datos] CHECK CONSTRAINT [ELEMENTOS_ELEM_PLANTILLA_FK]
GO
/****** Object:  ForeignKey [DATOS_ID_DATO_FK]    Script Date: 10/15/2012
02:08:52 ******/
ALTER TABLE [dbo].[Ordenacion]  WITH CHECK ADD  CONSTRAINT [DATOS_ID_DATO_FK]
FOREIGN KEY([ID_DATO])
REFERENCES [dbo].[Datos] ([ID_DATO])
GO
ALTER TABLE [dbo].[Ordenacion] CHECK CONSTRAINT [DATOS_ID_DATO_FK]
GO
/****** Object:  ForeignKey [PAGINAS_ID_PAG_FK]    Script Date: 10/15/2012
02:08:52 ******/
ALTER TABLE [dbo].[Ordenacion]  WITH CHECK ADD  CONSTRAINT [PAGINAS_ID_PAG_FK]
FOREIGN KEY([ID_PAG])
REFERENCES [dbo].[Paginas] ([ID_PAG])
GO
ALTER TABLE [dbo].[Ordenacion] CHECK CONSTRAINT [PAGINAS_ID_PAG_FK]
GO
/****** Object:  ForeignKey [PLANTILLAS_ELEM_PLANTILLA_FK]    Script Date:
10/15/2012 02:08:52 ******/
ALTER TABLE [dbo].[Ordenacion]  WITH CHECK ADD  CONSTRAINT
[PLANTILLAS_ELEM_PLANTILLA_FK] FOREIGN KEY([ELEM_PLANTILLA])
REFERENCES [dbo].[Elementos] ([ELEM_PLANTILLA])
GO
ALTER TABLE [dbo].[Ordenacion] CHECK CONSTRAINT [PLANTILLAS_ELEM_PLANTILLA_FK]
GO
```

Apéndice II. Código Web Service

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Services;
using System.Xml;

namespace ProyectoUC3
{
    /// <summary>
    /// Descripción breve de WebService1
    /// </summary>
    [WebService(Namespace = "http://tempuri.org/")]
    [WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
    [System.ComponentModel.ToolboxItem(false)]
    // Para permitir que se llame a este servicio Web desde un script, usando
ASP.NET AJAX, quite la marca de comentario de la línea siguiente.
    // [System.Web.Script.Services.ScriptService]
    public class WebService1 : System.Web.Services.WebService
    {

        [WebMethod] //Proporciona la versión actual del Menú en la base de
datos
        public string versionMenu()
        {
            UCEntities Test2 = new UCEntities();
            var valores = (from cont in Test2.Menu
                           select cont.Vers);
            var maximo = valores.Max();

            return maximo;
        }

        [WebMethod] //Carga el menú actual
        public XmlDocument getLatestMenu()
        {
            UCEntities context = new UCEntities();
            var query = (from row in context.Menu
                         orderby row.ID_MENU
                         select row);

            XmlDocument doc = new XmlDocument();

            XmlElement tabla_menus = doc.CreateElement("tabla_menus");
            doc.AppendChild(tabla_menus);


            foreach (var e in query)
            {
                XmlElement menu = doc.CreateElement("menu");
                tabla_menus.AppendChild(menu);


                XmlElement id_menu = doc.CreateElement("ID_MENU");
                id_menu.InnerText = e.ID_MENU.ToString();
                XmlElement texto = doc.CreateElement("Texto");
                texto.InnerText = e.Texto.ToString();
                XmlElement padre = doc.CreateElement("Padre");
                padre.InnerText = e.Padre.ToString();
                XmlElement prioridad = doc.CreateElement("Prioridad");
                prioridad.InnerText = e.Prioridad.ToString();
```

```
                XmlElement id_pag = doc.CreateElement("ID_PAG");
                id_pag.InnerText = e.ID_PAG.ToString();
                XmlElement vers = doc.CreateElement("Vers");
                vers.InnerText = e.Vers.ToString();
                menu.AppendChild(id_menu);
                menu.AppendChild(texto);
                menu.AppendChild(padre);
                menu.AppendChild(prioridad);
                menu.AppendChild(id_pag);
                menu.AppendChild(vers);
            }

            return doc;
        }

        [WebMethod]
        public XmlDocument getPage(int numPag)
        {
            UCEntities context = new UCEntities();
            var query = (from row in context.Paginas
                         where row.ID_PAG == numPag
                         select row.ID_PLANTILLA);

            int tipoPag = query.FirstOrDefault();

            XmlDocument doc = new XmlDocument();

            XmlElement paginaSolicitada =
doc.CreateElement("paginaSolicitada");
            doc.AppendChild(paginaSolicitada);

            XmlElement tipo = doc.CreateElement("tipo");
            paginaSolicitada.AppendChild(tipo);

            if (tipoPag == 6)
            {
                tipo.InnerText = "Mapa";
                XmlElement contenido = getContenidoMapa(context, numPag,
tipoPag, doc);
                paginaSolicitada.AppendChild(contenido);
            }
            else {
                if (tipoPag < 4)
                {
                    tipo.InnerText = "Ficha";
                }
                else if (tipoPag == 4)
                {
                    tipo.InnerText = "Texto";
                }
                else if (tipoPag == 5)
                {
                    tipo.InnerText = "TextoHTML";
                }
                XmlElement contenido = getContenidoPage(context, numPag,
tipoPag, doc);
                paginaSolicitada.AppendChild(contenido);
            }

            return doc;
        }

        public XmlElement getContenidoPage(UCEntities context, int numPag, int
tipoPag, XmlDocument doc)
        {
            XmlElement contenido = doc.CreateElement("contenido");
            var query2 = context.Ordenacion.Where(row => row.ID_PAG ==
numPag).Select(row => new { row.ID_DATO, row.ELEM_PLANTILLA });
```

```
            string tag;

            foreach (var e in query2)
            {

                var query3 = context.Elementos.Where(row => row.ELEM_PLANTILLA
== e.ELEM_PLANTILLA).Select(row => row.NOMBRE);
                var query4 = context.Elementos.Where(row => row.ELEM_PLANTILLA
== e.ELEM_PLANTILLA).Select(row => row.PADRE);
                var query5 = context.Elementos.Where(row => row.ELEM_PLANTILLA
== e.ELEM_PLANTILLA).Select(row => row.ELEM_PLANTILLA);
                tag = query3.FirstOrDefault();
                query3 = context.Datos.Where(row => row.ID_DATO ==
e.ID_DATO).Select(row => row.VALOR);


                if ((query4.FirstOrDefault() == null) || ((tipoPag == 1) &&
(query4.FirstOrDefault() < 20)) || ((tipoPag == 2) && (query4.FirstOrDefault()
< 30)) || ((tipoPag == 3) && (query4.FirstOrDefault() < 30)) || ((tipoPag ==
4) && (query4.FirstOrDefault() < 30)) || ((tipoPag == 5) &&
(query4.FirstOrDefault() < 30)))
                {

                    XmlElement tagLabel = doc.CreateElement(tag);
                    contenido.AppendChild(tagLabel);

                     if ((query5.FirstOrDefault()==8))
                     {

tagLabel.AppendChild(doc.CreateCDataSection(query3.FirstOrDefault()));
                    }
                     else
                     {
                        tagLabel.InnerText = query3.FirstOrDefault();
                    }
                }
            }
            return contenido;
        }

        public XmlElement getContenidoMapa(UCEntities context, int numPag, int
tipoPag, XmlDocument doc)
        {
            XmlElement localizaciones = doc.CreateElement("localizaciones");
            var query2 = context.Ordenacion.Where(row => row.ID_PAG ==
numPag).Select(row => new { row.ID_DATO, row.ELEM_PLANTILLA });
            string tag;
            int i = 0;
            XmlElement loc = doc.CreateElement("loc");

            foreach (var e in query2)
            {
                if (i % 2 == 0)
                {
                    loc = doc.CreateElement("loc");
                    localizaciones.AppendChild(loc);
                }
                var query3 = context.Elementos.Where(row => row.ELEM_PLANTILLA
== e.ELEM_PLANTILLA).Select(row => row.NOMBRE);
                tag = query3.FirstOrDefault();
                query3 = context.Datos.Where(row => row.ID_DATO ==
e.ID_DATO).Select(row => row.VALOR);

                XmlElement tagLabel = doc.CreateElement(tag);
                loc.AppendChild(tagLabel);
                tagLabel.InnerText = query3.FirstOrDefault();

                i = i + 1;
```

```
            }
            return localizaciones;
        }
    }
}
```

Apéndice III. Código aplicación iPhone (Principales clases)

```
//
//  KPAAppDelegate.h
//  UCProject
//
//  Created by Pablo Pérez on 21/09/12.
//  Copyright (c) 2012 __MyCompanyName__. All rights reserved.
//

#import <UIKit/UIKit.h>
#import <RestKit/RestKit.h>

@interface KPAAppDelegate : UIResponder <UIApplicationDelegate>{
    NSString *_serverURL;
}

@property (strong, nonatomic) UIWindow *window;

@property (readonly, strong, nonatomic) NSManagedObjectContext *managedObjectContext;
@property (readonly, strong, nonatomic) NSManagedObjectModel *managedObjectModel;
@property (readonly, strong, nonatomic) NSPersistentStoreCoordinator
*persistentStoreCoordinator;
@property (strong, nonatomic) NSString *serverURL;

- (void)saveContext;
- (void)showAlert:(NSString *)text;
- (NSURL *)applicationDocumentsDirectory;

@end
```

```
//
//  KPAAppDelegate.m
//  UCProject
//
//  Created by Pablo Pérez on 21/09/12.
//  Copyright (c) 2012 __MyCompanyName__. All rights reserved.
//

#import "KPAAppDelegate.h"
#import <RestKit/RestKit.h>
#import "modelo.h"
#import "KPAWaitViewController.h"

@implementation KPAAppDelegate

@synthesize window = _window;
@synthesize managedObjectContext = __managedObjectContext;
@synthesize managedObjectModel = __managedObjectModel;
@synthesize persistentStoreCoordinator = __persistentStoreCoordinator;
@synthesize serverURL=_serverURL;

- (BOOL)application:(UIApplication *)application
didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
    KPAWaitViewController *view=(KPAWaitViewController *)self.window.rootViewController;
    view.managedObjectContext=self.managedObjectContext;

    return YES;
}
//// OTHER DEFAULT APPDELEGATE METHODS
- (void)applicationWillResignActive:(UIApplication *)application{

}

- (void)applicationDidEnterBackground:(UIApplication *)application{

}

- (void)applicationWillEnterForeground:(UIApplication *)application{
```

```
}

- (void)applicationDidBecomeActive:(UIApplication *)application{

}

- (void)applicationWillTerminate:(UIApplication *)application{
    // Saves changes in the application's managed object context before the application
terminates.
    [self saveContext];
}

-(void)cargarVistaPrincipal{
    KPAWaitViewController *view=(KPAWaitViewController *)self.window.rootViewController;
    [view loadMenu];
}


#pragma mark - Core Data stack

// Returns the managed object context for the application.
// If the context doesn't already exist, it is created and bound to the persistent store
coordinator for the application.
- (NSManagedObjectContext *)managedObjectContext
{
    if (__managedObjectContext != nil) {
        return __managedObjectContext;
    }

    NSPersistentStoreCoordinator *coordinator = [self persistentStoreCoordinator];
    if (coordinator != nil) {
        __managedObjectContext = [[NSManagedObjectContext alloc] init];
        [__managedObjectContext setPersistentStoreCoordinator:coordinator];
    }
    return __managedObjectContext;
}

// Returns the managed object model for the application.
// If the model doesn't already exist, it is created from the application's model.
- (NSManagedObjectModel *)managedObjectModel
{
    if (__managedObjectModel != nil) {
        return __managedObjectModel;
    }
    NSURL *modelURL = [[NSBundle mainBundle] URLForResource:@"UCProject"
withExtension:@"momd"];
    __managedObjectModel = [[NSManagedObjectModel alloc]
initWithContentsOfURL:modelURL];
    return __managedObjectModel;
}

// Returns the persistent store coordinator for the application.
// If the coordinator doesn't already exist, it is created and the application's store
added to it.
- (NSPersistentStoreCoordinator *)persistentStoreCoordinator
{
    if (__persistentStoreCoordinator != nil) {
        return __persistentStoreCoordinator;
    }

    NSURL *storeURL = [[self applicationDocumentsDirectory]
URLByAppendingPathComponent:@"UCProject.sqlite"];

    NSError *error = nil;
    __persistentStoreCoordinator = [[NSPersistentStoreCoordinator alloc]
initWithManagedObjectModel:[self managedObjectModel]];
    if (![__persistentStoreCoordinator addPersistentStoreWithType:NSSQLiteStoreType
configuration:nil URL:storeURL options:nil error:&error]) {
        NSLog(@"Unresolved error %@, %@", error, [error userInfo]);
        abort();
    }

    return __persistentStoreCoordinator;
}
```

```
//// DOCUMENTS DIRECTORY
#pragma mark - Application's Documents directory

// Returns the URL to the application's Documents directory.
- (NSURL *)applicationDocumentsDirectory
{
    return [[[NSFileManager defaultManager] URLsForDirectory:NSDocumentDirectory
inDomains:NSUserDomainMask] lastObject];
}

//// SHOW ALERT
-(void) showAlert:(NSString *)text{
    UIAlertView *alert = [[UIAlertView alloc]
                          initWithTitle:@"Alerta"
                          message:text
                          delegate:self
                          cancelButtonTitle:@"Aceptar"
                          otherButtonTitles:nil];
    [alert show];
}

- (void)saveContext
{
    NSError *error = nil;
    NSManagedObjectContext *managedObjectContext = self.managedObjectContext;
    if (managedObjectContext != nil) {
        if ([managedObjectContext hasChanges] && ![managedObjectContext save:&error]) {
            NSLog(@"Unresolved error %@, %@", error, [error userInfo]);
            abort();
        }
    }
}

@end
```

```
//
//  KPAWaitViewController.h
//  UCProject
//
//  Created by Pablo Pérez on 22/09/12.
//  Copyright (c) 2012 __MyCompanyName__. All rights reserved.
//

#import <UIKit/UIKit.h>
#import <RestKit/RestKit.h>

@interface KPAWaitViewController :
UIViewController<NSURLConnectionDataDelegate,NSFetchedResultsControllerDelegate,RKReques
tDelegate,NSXMLParserDelegate>{
    NSMutableString *_reqOperation;
    NSString *_currentVersion;
    NSString *_latestVersion;
    NSMutableString *_nodeValue;
    RKClient *_client;
    NSMutableDictionary *_tmpMenuNode;
    NSArray *validNodes;
    NSMutableArray *_menus;
    NSString *_serverURL;
}
@property (weak, nonatomic) IBOutlet UIActivityIndicatorView *activityIndicator;
@property (nonatomic, retain) NSString *serverURL;
@property (strong, nonatomic) NSManagedObjectContext *managedObjectContext;
@property (nonatomic, retain) NSString *reqOperation;
@property (nonatomic, retain) NSString *currentVersion;
@property (nonatomic, retain) NSString *latestVersion;
@property (nonatomic, retain) NSString *nodeValue;
@property (nonatomic, retain) NSMutableArray *menus;
@property (nonatomic, retain) NSMutableDictionary *tmpMenuNode;
@property (nonatomic, retain) RKClient *client;
@property (weak, nonatomic) IBOutlet UILabel *infoLabel;

-(void)loadMenu;

@end
```

```
//
//  KPAWaitViewController.m
//  UCProject
//
//  Created by Pablo Pérez on 22/09/12.
//  Copyright (c) 2012 __MyCompanyName__. All rights reserved.
//

#import "KPAWaitViewController.h"
#import "KPAMainViewController.h"
#import "KPAAppDelegate.h"
#import "modelo.h"

@interface KPAWaitViewController ()

@end

@implementation KPAWaitViewController
@synthesize activityIndicator;
@synthesize serverURL=_serverURL;
@synthesize managedObjectContext = __managedObjectContext;
@synthesize reqOperation=_reqOperation;
@synthesize currentVersion=_currentVersion;
@synthesize nodeValue=_nodeValue;
@synthesize menus=_menus;
@synthesize tmpMenuNode=_tmpMenuNode;
@synthesize latestVersion=_latestVersion;
@synthesize client=_client;
@synthesize infoLabel = _infoLabel;

- (id)initWithNibName:(NSString *)nibNameOrNil bundle:(NSBundle *)nibBundleOrNil
{
    self = [super initWithNibName:nibNameOrNil bundle:nibBundleOrNil];
    if (self) {
        // Custom initialization
    }
    return self;
}

- (void)viewDidLoad
{
    [super viewDidLoad];
      // Do any additional setup after loading the view.
    if (self.managedObjectContext==nil) {
        KPAAppDelegate *delegate = (KPAAppDelegate *)[[UIApplication sharedApplication]
delegate];
        self.managedObjectContext=delegate.managedObjectContext;
    }

    [self.activityIndicator startAnimating];
    [self startApp];

    //Initial values
    self.tmpMenuNode =[NSMutableDictionary new];
    validNodes=[[NSArray alloc]
initWithObjects:@"ID_MENU",@"Texto",@"Padre",@"Prioridad",@"ID_PAG",@"Vers", nil];
    self.menus=[NSMutableArray new];

    self.currentVersion=[self getMenuVersion]; //1.0.1=Yes
}

- (void)viewDidUnload
{
    [self setActivityIndicator:nil];
    [self setInfoLabel:nil];
    [super viewDidUnload];
    // Release any retained subviews of the main view.
}

-
(BOOL)shouldAutorotateToInterfaceOrientation:(UIInterfaceOrientation)interfaceOrientatio
n
{
    return (interfaceOrientation == UIInterfaceOrientationPortrait);
}
```

```
//LOAD MENU
-(void)loadMenu{
    [self performSegueWithIdentifier:@"showApp" sender:self];
}


//// SHOW ALERT
-(void) showAlert:(NSString *)text{
    UIAlertView *alert = [[UIAlertView alloc]
                          initWithTitle:@"Alerta"
                          message:text
                          delegate:self
                          cancelButtonTitle:@"Aceptar"
                          otherButtonTitles:nil];
    [alert show];
}

-(void) getServerIP{
    UIAlertView* dialog = [[UIAlertView alloc] initWithTitle:@"Dirección del servidor"
                                                     message:@"  "
                                                    delegate:self
                                           cancelButtonTitle:@"Cancel"
                                           otherButtonTitles:@"OK", nil];

    UITextField *nameField = [[UITextField alloc]
                              initWithFrame:CGRectMake(20.0, 45.0, 245.0, 25.0)];
    [nameField setBackgroundColor:[UIColor whiteColor]];
    [nameField setText:@"192.168.1.143:8070"]; // Valor por defecto. Otras pruebas:
kpa.hopto.org:8070
    [dialog addSubview:nameField];
    [dialog show];
}

- (void) alertView:(UIAlertView *)alertView clickedButtonAtIndex:(NSInteger)buttonIndex
{
    if (buttonIndex != 0)  // 0 == the cancel button
    {
        for (UIView* view in alertView.subviews)
        {
            if ([view isKindOfClass:[UITextField class]])
            {
                UITextField* textField = (UITextField*)view;
                self.serverURL=textField.text;
                KPAAppDelegate *delegate = (KPAAppDelegate *)[[UIApplication
sharedApplication] delegate];
                delegate.serverURL=textField.text;
                [self checkConnection:textField.text];
                break;
            }
        }
    }else {
        [self.activityIndicator stopAnimating];
        //self.infoLabel.text=@"Operación cancelada por el usuario";
    }
}

//1. COMPROBAR CONEXION CON SERVIDOR
-(void)checkConnection:(NSString *)url{
    self.infoLabel.text=@"Comprobando la conexión con el servidor";
    NSURLRequest *req=[NSURLRequest requestWithURL:[NSURL URLWithString:[NSString
stringWithFormat:@"http://%@",url]] cachePolicy:NSURLRequestReloadIgnoringCacheData
timeoutInterval:20.0];

    NSURLConnection *con = [NSURLConnection connectionWithRequest:req delegate:self];
    [con start];
}

-(void)connectionDidFinishLoading:(NSURLConnection *)connection{
    [self initClient];
    [self checkMenuVersion];
}

-(void)connection:(NSURLConnection *)connection didFailWithError:(NSError *)error{
    [self showAlert:[NSString stringWithFormat:@"Error en la conexión con el servidor:
%@",[error localizedDescription]]];
```

```
    [self.activityIndicator stopAnimating];
    self.infoLabel.text=[NSString stringWithFormat:@"Error en la conexión con http://%@.
Revise los datos de conexión e inténtelo de nuevo más tarde.",self.serverURL];
}

//// WEB SERVICES
-(void)initClient{
    self.client = [RKClient clientWithBaseURLString:[NSString
stringWithFormat:@"http://%@/proyecto/",self.serverURL]];
}

-(void)checkMenuVersion{
    self.reqOperation=@"checkMenuVersion";
    self.infoLabel.text=[NSString stringWithFormat:@"Comprobando la versión del menú
(%@)",self.currentVersion];

    NSDictionary *diccio = [NSDictionary
dictionaryWithKeysAndObjects:@"versDevice",self.currentVersion, nil];
    [self.client get:@"/WebService1.asmx/versionMenu" queryParameters:diccio
delegate:self];
}

-(void)getLatestMenu{
    self.reqOperation=@"getLatestMenu";
    self.infoLabel.text=@"Descargando nueva versión del menú";

    [self.client get:@"/WebService1.asmx/getLatestMenu" delegate:self];
}

#pragma mark RKRequestDelegate methods
- (void)request:(RKRequest*)request didLoadResponse:(RKResponse*)response {
    if ([request isGET]) {
        [self parseMenuData:[response bodyAsString]];
    }else if ([response isNotFound])
    {       // No ha devuelto nada
        [self showAlert:[NSString stringWithFormat:@"El recurso con ruta '%@' no ha sido
encontrado.", [request resourcePath]]];
    }else{
        [self showAlert:[response bodyAsString]];
    }
}

//// XMLREADER
-(void)parseMenuData:(NSString *)menuData{
    NSData *data=[menuData dataUsingEncoding:NSUTF8StringEncoding];
    NSXMLParser *parser = [[NSXMLParser alloc] initWithData:data];

    [parser setDelegate:self];
    [parser parse];
}


-(void)parser:(NSXMLParser *)parser parseErrorOccurred:(NSError *)parseError{
    NSLog(@"Parser error:%@",parseError);
}

-(void)parser:(NSXMLParser *)parser didStartElement:(NSString *)elementName
namespaceURI:(NSString *)namespaceURI qualifiedName:(NSString *)qName
attributes:(NSDictionary *)attributeDict{
    self.nodeValue=@"";
}

-(void)parser:(NSXMLParser *)parser didEndElement:(NSString *)elementName
namespaceURI:(NSString *)namespaceURI qualifiedName:(NSString *)qName{
    if([self.reqOperation isEqualToString:@"getLatestMenu"]){
        if ([elementName isEqualToString:@"menu"]) {
            NSDictionary *tmp=[[NSDictionary alloc]
initWithDictionary:self.tmpMenuNode];
            [self.menus addObject:tmp];
        }else if ([validNodes containsObject:elementName]) {
            [self.tmpMenuNode setValue:self.nodeValue forKey:elementName];
        }
    }else if([self.reqOperation isEqualToString:@"checkMenuVersion"]){
        self.latestVersion=self.nodeValue;
    }
    self.nodeValue=@"";
```

```
}
-(void)parserDidEndDocument:(NSXMLParser *)parser{
    if ([self.reqOperation isEqualToString:@"checkMenuVersion"]){
        if ([self.latestVersion isEqualToString:self.currentVersion]) {
            //TRUE: Cargar menu de CoreData
            [self loadMenu];
        }else {
            //FALSE: descargar ultima version
            [self getLatestMenu];
        }
    }else if([self.reqOperation isEqualToString:@"getLatestMenu"]){
        [self saveMenuItems:self.menus];
        [self loadMenu];
    }
}

-(void)parser:(NSXMLParser *)parser foundCharacters:(NSString *)string{
    self.nodeValue=[NSString stringWithFormat:@"%@%@",self.nodeValue,string];
}

//// COREDATA
// >> COREDATA - menuVersion
-(NSString *)getMenuVersion{
    NSArray *fetchedObjects = [self fetchVersion];

    if (fetchedObjects.count>0) {
        Version *mv=[fetchedObjects objectAtIndex:0];
        return mv.number;
    }else{
        [self saveMenuVersion:@"0.0.0"];
        return @"0.0.0";
    }
}

-(NSArray *)fetchVersion{
    NSFetchRequest *fetchRequest = [[NSFetchRequest alloc] init];
    NSEntityDescription *entity = [NSEntityDescription entityForName:@"Version"

inManagedObjectContext:self.managedObjectContext];
    [fetchRequest setEntity:entity];

    NSError *error = nil;
    NSArray *fetchedObjects = [self.managedObjectContext
executeFetchRequest:fetchRequest error:&error];
    if(error!=nil){NSLog(@"Error fetching data: %@",error);}
    return fetchedObjects;
}

-(void)changeMenuVersion:(NSString *)newVersion{
    NSFetchRequest *fetchRequest = [[NSFetchRequest alloc] init];
    NSEntityDescription *entity = [NSEntityDescription entityForName:@"Version"

inManagedObjectContext:self.managedObjectContext];
    [fetchRequest setEntity:entity];

    NSError *error = nil;
    NSArray *fetchedObjects = [self.managedObjectContext
executeFetchRequest:fetchRequest error:&error];

    if(error!=nil){NSLog(@"Error fetching data: %@",error);}

    Version *mv=[fetchedObjects objectAtIndex:0];
    mv.number=newVersion;

    // Save the context.
    [self saveContext];
}


-(void)saveMenuVersion:(NSString *)menuVersion{
    NSEntityDescription *nuevaVersion = [NSEntityDescription
insertNewObjectForEntityForName:@"Version"
inManagedObjectContext:self.managedObjectContext];
    [nuevaVersion setValue:menuVersion forKey:@"number"];

    // Save the context.
```

```
    [self saveContext];
}

// >> COREDATA - menuItems
-(void)saveMenuItems:(NSMutableArray *)menuArray{
    //Save New Version
    [self changeMenuVersion:self.latestVersion];
    Version *nuevaVersion=[[self fetchVersion] objectAtIndex:0];

    //Delete previous menuItems
    [self deleteAllMenuItems];

    //Save Menu Items
    for (id d in menuArray){
        MenuItem *nuevoMenuItem=[NSEntityDescription
insertNewObjectForEntityForName:@"MenuItem"
inManagedObjectContext:self.managedObjectContext];

        NSNumberFormatter * f = [[NSNumberFormatter alloc] init];
        [f setNumberStyle:NSNumberFormatterDecimalStyle];

        [nuevoMenuItem setValue:[f numberFromString:[d objectForKey:@"ID_MENU"]]
forKey:@"id_menu"];
        [nuevoMenuItem setValue:[d objectForKey:@"Texto"] forKey:@"texto"];
        [nuevoMenuItem setValue:[f numberFromString:[d objectForKey:@"Padre"]]
forKey:@"padre"];
        [nuevoMenuItem setValue:[f numberFromString:[d objectForKey:@"Prioridad"]]
forKey:@"prioridad"];
        [nuevoMenuItem setValue:[f numberFromString:[d objectForKey:@"ID_PAG"]]
forKey:@"id_pag"];

        [nuevaVersion addContieneObject:nuevoMenuItem];
    }

    // Save the context.
    [self saveContext];
}

-(void)deleteAllMenuItems{
    NSFetchRequest *fetchRequest = [[NSFetchRequest alloc] init];
    NSEntityDescription *entity = [NSEntityDescription entityForName:@"MenuItem"

inManagedObjectContext:self.managedObjectContext];
    [fetchRequest setEntity:entity];
    [fetchRequest setIncludesPropertyValues:NO]; //only fetch the managedObjectID

    NSError *error = nil;
    NSArray *fetchedObjects = [self.managedObjectContext
executeFetchRequest:fetchRequest error:&error];
    if(error!=nil){NSLog(@"Error fetching data: %@",error);}

    for (NSManagedObject * item in fetchedObjects) {
        [self.managedObjectContext deleteObject:item];
    }
    // Save the context.
    [self saveContext];
}

// >> COREDATA - default methods
- (void)saveContext
{
    NSError *error = nil;
    NSManagedObjectContext *managedObjectContext = self.managedObjectContext;
    if (managedObjectContext != nil) {
        if ([managedObjectContext hasChanges] && ![managedObjectContext save:&error]) {
            NSLog(@"Unresolved error %@, %@", error, [error userInfo]);
            abort();
        }
    }
}



@end
```

```
//
//  KPAMainViewController.h
//  UCProject
//
//  Created by Pablo Pérez on 22/09/12.
//  Copyright (c) 2012 __MyCompanyName__. All rights reserved.
//

#import <UIKit/UIKit.h>
#import <CoreData/CoreData.h>
#import <RestKit/RestKit.h>

@interface KPAMainViewController :
UITableViewController<NSFetchedResultsControllerDelegate,RKRequestDelegate,NSXMLParserDe
legate>{
    NSNumber *_padre;
    NSString *_padreTexto;
    NSNumber *_subMenu;
    NSArray *_nodes;
    RKClient *_client2;
    NSMutableString *_nodeValue;
    NSMutableDictionary *_pagDict;
    NSMutableArray *_pagArray;
    NSMutableString *tipo;
    NSString *_serverURL;
}

@property (strong, nonatomic) NSManagedObjectContext *managedObjectContext;
@property (strong, nonatomic) NSString *serverURL;
@property (strong, nonatomic) NSNumber *padre;
@property (strong, nonatomic) NSString *padreTexto;
@property (strong, nonatomic) NSNumber *subMenu;
@property (strong, nonatomic) NSArray *nodes;
@property (strong, nonatomic) RKClient *client2;
@property (nonatomic, retain) NSString *nodeValue;
@property (nonatomic, retain) NSMutableDictionary *pagDict;
@property (nonatomic, retain) NSMutableArray *pagArray;

@end
```

```
//
//  KPAMainViewController.m
//  UCProject
//
//  Created by Pablo Pérez on 22/09/12.
//  Copyright (c) 2012 __MyCompanyName__. All rights reserved.
//

#import "KPAMainViewController.h"
#import "KPAAppDelegate.h"
#import "modelo.h"
#import <RestKit/RestKit.h>
#import "KPAFichaViewController.h"
#import "KPAMapaViewController.h"
#import "KPATextoHTMLViewController.h"

@interface KPAMainViewController ()

@end

@implementation KPAMainViewController

@synthesize managedObjectContext = __managedObjectContext;
@synthesize padre=_padre;
@synthesize padreTexto=_padreTexto;
@synthesize subMenu=_subMenu;
@synthesize nodes=_nodes;
@synthesize client2=_client2;
@synthesize nodeValue=_nodeValue;
@synthesize pagDict=_pagDict;
@synthesize pagArray=_pagArray;
@synthesize serverURL=_serverURL;

- (id)initWithStyle:(UITableViewStyle)style
{
```

```
    self = [super initWithStyle:style];
    if (self) {
        // Custom initialization
    }
    return self;
}

- (void)viewDidLoad
{
    [super viewDidLoad];

    //Parameters from AppDelegate -- managedObjectContext,serverURL
    KPAAppDelegate *delegate = (KPAAppDelegate *)[[UIApplication sharedApplication]
delegate];
    if (self.managedObjectContext==nil) {
        self.managedObjectContext=delegate.managedObjectContext;
    }
    if (self.serverURL==nil) {
        self.serverURL=delegate.serverURL;
    }

    if (self.padreTexto==nil) {
         [self.navigationItem setTitle:@"Inicio"];
    }else{
         [self.navigationItem setTitle:self.padreTexto];
    }
    if (self.subMenu==nil) {
        self.subMenu=[NSNumber numberWithInt:0];
    }
    self.pagDict = [NSMutableDictionary new];
    self.pagArray = [NSMutableArray new];
    tipo=[NSMutableString stringWithString:@""];

    //Initial opertions
    if(self.nodes== nil){
        [self loadMenu];
    }
}

- (void)viewDidUnload
{
    [super viewDidUnload];
    // Release any retained subviews of the main view.
}

-
(BOOL)shouldAutorotateToInterfaceOrientation:(UIInterfaceOrientation)interfaceOrientatio
n
{
    return (interfaceOrientation == UIInterfaceOrientationPortrait);
}

//// LOAD MENU
-(void)loadMenu{
    // Obtener los elementos que no tienen padre ordenados por prioridad
    if (self.padre==nil) {
        self.nodes=[self getElementsWithPredicate:@"padre==nil"];
    }else{
        self.nodes=[self getElementsWithPredicate:[NSString
stringWithFormat:@"padre==%d",[self.padre intValue]]];
    }
}


-(NSArray *)getElementsWithPredicate:(NSString *)txt{
    // Ordenar siempre por prioridad
    NSFetchRequest *fetchRequest = [[NSFetchRequest alloc] init];
    NSEntityDescription *entity = [NSEntityDescription entityForName:@"MenuItem"

inManagedObjectContext:self.managedObjectContext];
    [fetchRequest setEntity:entity];

    NSPredicate *predicate = [NSPredicate predicateWithFormat:[NSString
stringWithFormat:@"(%@)",txt]];
    NSSortDescriptor *descriptor = [NSSortDescriptor sortDescriptorWithKey:@"prioridad"
                                                                 ascending:YES];
```

```
    [fetchRequest setPredicate:predicate];
    [fetchRequest setSortDescriptors:[[NSArray alloc] initWithObjects:descriptor, nil]];

    NSError *error = nil;
    NSArray *fetchedObjects = [self.managedObjectContext
executeFetchRequest:fetchRequest error:&error];

    if(error!=nil){NSLog(@"Error fetching data: %@",error);}

    return fetchedObjects;
}


//// TABLE
#pragma mark - Table view data source

- (NSInteger)numberOfSectionsInTableView:(UITableView *)tableView
{
    // Return the number of sections.
    return 1;
}

- (NSInteger)tableView:(UITableView *)tableView numberOfRowsInSection:(NSInteger)section
{
    // Return the number of rows in the section.
    return self.nodes.count;
}

- (UITableViewCell *)tableView:(UITableView *)tableView
cellForRowAtIndexPath:(NSIndexPath *)indexPath
{
    UITableViewCell *cell = [tableView dequeueReusableCellWithIdentifier:@"Cell"];

    if (cell == nil) {
        cell = [[UITableViewCell alloc] initWithStyle:UITableViewCellStyleSubtitle
reuseIdentifier:@"Cell"];
    }

    // Configure the cell...
    MenuItem *m = [self.nodes objectAtIndex:indexPath.row];
    cell.textLabel.text =  m.texto;
    cell.accessoryType = UITableViewCellAccessoryDisclosureIndicator;

    return cell;
}

#pragma mark - Table view delegate

- (void)tableView:(UITableView *)tableView didSelectRowAtIndexPath:(NSIndexPath
*)indexPath
{
    MenuItem *selectedItem=[self.nodes objectAtIndex:indexPath.row];
    if ([self.subMenu intValue]==4) {
        @try {
            [self getPage:selectedItem.id_pag];
        }
        @catch (NSException *exception) {
            NSLog(@"Error cargando la pagina:%@",[exception description]);
        }
        @finally {
             NSLog(@"ULTIMO MENU!!!");
        }

    }else if ([selectedItem.id_pag intValue]==0) {
        [self performSegueWithIdentifier:[NSString
stringWithFormat:@"showMenu%d",[self.subMenu intValue]+1] sender:self];
    }else{
        [self getPage:selectedItem.id_pag];
    }
}

//// REQUEST PAGE
-(void)getPage:(NSNumber *)pageID{
    NSLog(@"Device:%@",[UIDevice currentDevice].model);
    //Request page
```

```
    NSDictionary *diccio = [NSDictionary
dictionaryWithKeysAndObjects:@"numPag",[NSString stringWithFormat:@"%@",pageID], nil];

    self.client2 = [RKClient clientWithBaseURLString:[NSString
stringWithFormat:@"http://%@/proyecto/",self.serverURL]];
    [self.client2 get:@"/WebService1.asmx/getPage" queryParameters:diccio
delegate:self];
}

//// WEB SERVICES
#pragma mark RKRequestDelegate methods
- (void)request:(RKRequest*)request didLoadResponse:(RKResponse*)response {
    KPAAppDelegate *delegate = (KPAAppDelegate *)[[UIApplication sharedApplication]
delegate];

    if ([request isGET]) {
        [self parseMenuData:[response bodyAsString]];
    }else if ([response isNotFound])
    {       // No ha devuelto nada
        [delegate showAlert:[NSString stringWithFormat:@"El recurso con ruta '%@' no ha
sido encontrado.", [request resourcePath]]];
    }else{
        [delegate showAlert:[response bodyAsString]];
    }
}

//// XMLREADER
-(void)parseMenuData:(NSString *)menuData{
    NSData *data=[menuData dataUsingEncoding:NSUTF8StringEncoding];
    NSXMLParser *parser = [[NSXMLParser alloc] initWithData:data];
    [parser setDelegate:self];
    [parser parse];
}


-(void)parser:(NSXMLParser *)parser parseErrorOccurred:(NSError *)parseError{
    NSLog(@"Parser error:%@",parseError);
}

-(void)parser:(NSXMLParser *)parser didStartElement:(NSString *)elementName
namespaceURI:(NSString *)namespaceURI qualifiedName:(NSString *)qName
attributes:(NSDictionary *)attributeDict{
    if([elementName isEqualToString:@"localizaciones"]){
        // Si el XML contiene el tag <localizaciones> se trata de un mapa
        self.pagArray=[NSMutableArray new];
    }

    self.nodeValue=@"";
}

-(void)parser:(NSXMLParser *)parser didEndElement:(NSString *)elementName
namespaceURI:(NSString *)namespaceURI qualifiedName:(NSString *)qName{

    // Cada tipo de pag se parsea de una manera distinta; 3 tipos: Ficha, Mapa,
TextoHTML
    if([elementName isEqualToString:@"tipo"]){
        tipo=[NSMutableString stringWithString:self.nodeValue];
        [self.pagDict setValue:self.nodeValue forKey:elementName];
    } else if ([elementName isEqualToString:@"loc"]){ //Tipo Mapa
        NSArray *keys = [[NSArray alloc]
initWithObjects:@"Nombre_Mapa",@"Latitud",@"Longitud",nil];
        NSString *coords= [self.pagDict objectForKey:@"Coord_Mapa"];
        NSRange range = [coords rangeOfString:@","];
        NSArray *objects = [[NSArray alloc] initWithObjects:[self.pagDict
objectForKey:@"Nombre_Mapa"],[coords substringToIndex:range.location],[coords
substringFromIndex:range.location+1],nil];
        NSDictionary *tmp=[[NSDictionary alloc] initWithObjects:objects forKeys:keys];
        [self.pagArray addObject:tmp];
    }else if ([elementName isEqualToString:@"paginaSolicitada"]==NO && [elementName
isEqualToString:@"contenido"]==NO) {
        NSRange range = [elementName rangeOfString:@"_"];
        if (range.location==NSNotFound || [tipo isEqualToString:@"Mapa"]){
            [self.pagDict setValue:self.nodeValue forKey:elementName];
        }else{
            [self.pagDict setValue:self.nodeValue forKey:[elementName
substringToIndex:range.location]];
```

```
            [self.pagDict setValue:[elementName substringFromIndex:range.location+1]
forKey:@"subtipo"];
        }
    }
    self.nodeValue=@"";
}
-(void)parserDidEndDocument:(NSXMLParser *)parser{
    if ([tipo isEqualToString:@"Mapa"]) {
        [self cargarMapa:[NSArray arrayWithArray:self.pagArray]];
    }else if ([tipo isEqualToString:@"Ficha"]) {
        [self cargarFicha:[NSDictionary dictionaryWithDictionary:self.pagDict]];
    }else if ([tipo isEqualToString:@"TextoHTML"]) {
        [self cargarTextoHTML:[NSDictionary dictionaryWithDictionary:self.pagDict]];
    }
    tipo=[NSMutableString stringWithString:@""];//Se resetea la variable
}

-(void)parserDidStartDocument:(NSXMLParser *)parser{
    self.pagDict = [NSMutableDictionary new];
    self.pagArray = [NSMutableArray new];
}

-(void)parser:(NSXMLParser *)parser foundCharacters:(NSString *)string{
    self.nodeValue=[NSString stringWithFormat:@"%@%@",self.nodeValue,string];
}

//// LOAD PAGE:
-(void)cargarFicha:(NSDictionary *)page{
    NSIndexPath *indexPath = [self.tableView indexPathForSelectedRow];
    MenuItem *selectedItem=[self.nodes objectAtIndex:indexPath.row];

    //Transition to new Page
    UIStoryboard *storyboard = [UIStoryboard storyboardWithName:@"MainStoryboard"
bundle:nil];
    KPAFichaViewController *nuevaPagina = (KPAFichaViewController *)[storyboard
instantiateViewControllerWithIdentifier:@"Ficha"];
    nuevaPagina.pageContents=page;
    nuevaPagina.nombrePag=selectedItem.texto;
    [self.navigationController pushViewController:nuevaPagina animated:YES];
}

// Las pags Texto se guardan como un
diccionario:{"tipo":"TextoHTML","Titulo":"Informacion","Cuerpo:":<contenido HTML>}
-(void)cargarTextoHTML:(NSDictionary *)contenido{
    //Transition to new Page
    UIStoryboard *storyboard = [UIStoryboard storyboardWithName:@"MainStoryboard"
bundle:nil];
    KPATextoHTMLViewController *nuevaPagina = (KPATextoHTMLViewController *)[storyboard
instantiateViewControllerWithIdentifier:@"TextoHTML"];
    nuevaPagina.htmlContent=[contenido objectForKey:@"Cuerpo"];
    [self.navigationController pushViewController:nuevaPagina animated:YES];
}

// Los mapas se guardan como un array de diccionarios:
[{"Nombre_Mapa":"Medicina","Latitud":"43.57383","Longitud":"-
3.829304"},{"Nombre_Mapa"...},...]
-(void)cargarMapa:(NSArray *)mapaArray{
    NSIndexPath *indexPath = [self.tableView indexPathForSelectedRow];
    MenuItem *selectedItem=[self.nodes objectAtIndex:indexPath.row];

    //Transition to new Page
    UIStoryboard *storyboard = [UIStoryboard storyboardWithName:@"MainStoryboard"
bundle:nil];
    KPAMapaViewController *nuevaPagina = (KPAMapaViewController *)[storyboard
instantiateViewControllerWithIdentifier:@"Mapa"];
    nuevaPagina.mapaContents=mapaArray;
    nuevaPagina.nombreMapa=selectedItem.texto;
    [self.navigationController pushViewController:nuevaPagina animated:YES];
}

//// SEGUES
- (void)prepareForSegue:(UIStoryboardSegue *)segue sender:(id)sender
{
    if ([[segue identifier] isEqualToString:[NSString
stringWithFormat:@"showMenu%d",[self.subMenu intValue]+1]]) {
        NSIndexPath *indexPath = [self.tableView indexPathForSelectedRow];
```

```
        MenuItem *selectedItem=[self.nodes objectAtIndex:indexPath.row];

        [[segue destinationViewController] setPadre:selectedItem.id_menu];
        [[segue destinationViewController] setPadreTexto:selectedItem.texto];
        [[segue destinationViewController] setSubMenu:[NSNumber
numberWithInt:[self.subMenu intValue]+1]];
    }
}


@end
```

```
//
//  KPANoticiasViewController.h
//  UCProject
//
//  Created by Pablo Pérez on 12/10/12.
//  Copyright (c) 2012 __MyCompanyName__. All rights reserved.
//

#import <UIKit/UIKit.h>

@interface KPANoticiasViewController : UITableViewController<NSXMLParserDelegate>{
    NSMutableArray *_allEntries;
    NSMutableString *_nodeValue;
    NSMutableDictionary *tmpEntry;
    NSArray *validNodes;
}

@property (retain) NSMutableArray *allEntries;
@property (retain) NSString *nodeValue;

@end
```

```
//
//  KPANoticiasViewController.m
//  UCProject
//
//  Created by Pablo Pérez on 12/10/12.
//  Copyright (c) 2012 __MyCompanyName__. All rights reserved.
//

#import "KPANoticiasViewController.h"
#import "KPANoticiasDetalleViewController.h"
#import "RSSEntry.h"

#define CELL_CONTENT_WIDTH 320.0f
#define CELL_CONTENT_MARGIN 10.0f

@interface KPANoticiasViewController ()

@end

@implementation KPANoticiasViewController
@synthesize allEntries = _allEntries;
@synthesize nodeValue=_nodeValue;

- (id)initWithStyle:(UITableViewStyle)style
{
    self = [super initWithStyle:style];
    if (self) {
        // Custom initialization
    }
    return self;
}

- (void)viewDidLoad
{
    [super viewDidLoad];
    self.allEntries = [NSMutableArray array];
    validNodes=[[NSArray alloc]
initWithObjects:@"title",@"link",@"description",@"pubDate",@"guid", nil];
    [self getRSSFeed];
}
```

```
- (void)viewDidUnload
{
    [super viewDidUnload];
    // Release any retained subviews of the main view.
    // e.g. self.myOutlet = nil;
}


-
(BOOL)shouldAutorotateToInterfaceOrientation:(UIInterfaceOrientation)interfaceOrientatio
n
{
    return (interfaceOrientation == UIInterfaceOrientationPortrait);
}

//RSS
-(void)getRSSFeed{
    NSURLRequest *request = [NSURLRequest requestWithURL:[NSURL
URLWithString:@"http://www.unican.es/WebUC/Internet/Noticias_y_novedades/rss.xml"]];
    NSData *data = [NSURLConnection sendSynchronousRequest:request returningResponse:nil
error:nil];
    [self parseFeedData:data];
}

-(void)addEntrytoList:(NSDictionary *)entryDict{
    NSDateFormatter *dateFormat = [[NSDateFormatter alloc] init];
    [dateFormat setDateFormat:@"dd MMM yyyy HH:mm:ss ZZ"];
    NSDate *date = [dateFormat dateFromString:[entryDict objectForKey:@"pubDate"]];

    RSSEntry *newEntry = [[RSSEntry alloc] initWithEntryTitle:[entryDict
objectForKey:@"title"] entryLink:[entryDict objectForKey:@"link"]
entryDescription:[entryDict objectForKey:@"description"] entryPubDate:date
entryGuid:[entryDict objectForKey:@"guid"]];

    [self.allEntries insertObject:newEntry atIndex:[self.allEntries count]];
}

//// XMLREADER
-(void)parseFeedData:(NSData *)feedData{
    NSXMLParser *parser = [[NSXMLParser alloc] initWithData:feedData];

    [parser setDelegate:self];
    [parser parse];
}


-(void)parser:(NSXMLParser *)parser parseErrorOccurred:(NSError *)parseError{
    NSLog(@"Parser error:%@",parseError);
}

-(void)parser:(NSXMLParser *)parser didStartElement:(NSString *)elementName
namespaceURI:(NSString *)namespaceURI qualifiedName:(NSString *)qName
attributes:(NSDictionary *)attributeDict{
    if ([elementName isEqualToString:@"item"]) {
        tmpEntry=[NSMutableDictionary new];
    }
    self.nodeValue=@"";
}

-(void)parser:(NSXMLParser *)parser didEndElement:(NSString *)elementName
namespaceURI:(NSString *)namespaceURI qualifiedName:(NSString *)qName{
    if ([elementName isEqualToString:@"item"]) {
        [self addEntrytoList:[NSDictionary dictionaryWithDictionary:tmpEntry]];
    }else if ([validNodes containsObject:elementName]) {
        [tmpEntry setValue:self.nodeValue forKey:elementName];
    }
    self.nodeValue=@"";
}

-(void)parser:(NSXMLParser *)parser foundCharacters:(NSString *)string{
    self.nodeValue=[NSString stringWithFormat:@"%@%@",self.nodeValue,string];
}

//TABLA
#pragma mark - Table view data source
```

```
- (NSInteger)numberOfSectionsInTableView:(UITableView *)tableView
{
    // Return the number of sections.
    return 1;
}

- (NSInteger)tableView:(UITableView *)tableView numberOfRowsInSection:(NSInteger)section
{
    // Return the number of rows in the section.
    return [self.allEntries count];
}

- (UITableViewCell *)tableView:(UITableView *)tableView
cellForRowAtIndexPath:(NSIndexPath *)indexPath
{
    UITableViewCell *cell = [tableView dequeueReusableCellWithIdentifier:@"Cell"];
    if (cell == nil) {
        cell = [[UITableViewCell alloc] initWithStyle:UITableViewCellStyleSubtitle
reuseIdentifier:@"Cell"];
    }

    RSSEntry *entry = [self.allEntries objectAtIndex:indexPath.row];

    NSDateFormatter * dateFormatter = [[NSDateFormatter alloc] init];
    [dateFormatter setDateFormat:@"dd/MM/yyyy HH:mm"];
    NSString *entryDateString = [dateFormatter stringFromDate:entry.entryPubDate];

    cell.textLabel.text = [NSString stringWithFormat:@"%@", entryDateString];
    cell.detailTextLabel.text = entry.entryTitle;
    cell.detailTextLabel.numberOfLines=0;

    cell.textLabel.font = [UIFont fontWithName:@"Verdana" size:11];
    cell.detailTextLabel.font = [UIFont fontWithName:@"Verdana-Bold" size:12];
    cell.detailTextLabel.lineBreakMode=UILineBreakModeWordWrap;
    cell.detailTextLabel.textColor = [UIColor darkGrayColor];

    return cell;
}

-(CGFloat)tableView:(UITableView *)tableView heightForRowAtIndexPath:(NSIndexPath
*)indexPath{
    RSSEntry *entry = [self.allEntries objectAtIndex:indexPath.row];

    CGSize constraint = CGSizeMake(CELL_CONTENT_WIDTH - (CELL_CONTENT_MARGIN * 2),
20000.0f);
    CGSize size = [entry.entryTitle sizeWithFont:[UIFont fontWithName:@"Verdana-Bold"
size:12] constrainedToSize:constraint lineBreakMode:UILineBreakModeWordWrap];

    CGFloat height = MAX(size.height, 44.0f);

    return height + (CELL_CONTENT_MARGIN * 2);
}

#pragma mark - Table view delegate

- (void)tableView:(UITableView *)tableView didSelectRowAtIndexPath:(NSIndexPath
*)indexPath
{
    RSSEntry *entry = [self.allEntries objectAtIndex:indexPath.row];
    NSDateFormatter * dateFormatter = [[NSDateFormatter alloc] init];
    [dateFormatter setDateFormat:@"dd/MM/yyyy HH:mm"];
    NSString *entryDateString = [dateFormatter stringFromDate:entry.entryPubDate];

    UIStoryboard *storyboard = [UIStoryboard storyboardWithName:@"MainStoryboard"
bundle:nil];
    KPANoticiasDetalleViewController *nuevaPagina = (KPANoticiasDetalleViewController
*)[storyboard instantiateViewControllerWithIdentifier:@"DetalleNoticia"];
    nuevaPagina.ntitle=entry.entryTitle;
    nuevaPagina.description=[entry.entryDescription
stringByReplacingOccurrencesOfString:@"\n" withString:@""];
    nuevaPagina.link=entry.entryLink;
    nuevaPagina.pubDate=entryDateString;
    [self.navigationController pushViewController:nuevaPagina animated:YES];
}
```

```
@end
```

```
//
//  KPANoticiasDetalleViewController.h
//  UCProject
//
//  Created by Pablo Pérez on 13/10/12.
//  Copyright (c) 2012 __MyCompanyName__. All rights reserved.
//

#import <UIKit/UIKit.h>

@interface KPANoticiasDetalleViewController : UITableViewController{
    NSString *_ntitle;
    NSString *_pubDate;
    NSString *_description;
    NSString *_link;
}

@property (retain) NSString *ntitle;
@property (retain) NSString *pubDate;
@property (retain) NSString *description;
@property (retain) NSString *link;

@end
```

```
//
//  KPANoticiasDetalleViewController.m
//  UCProject
//
//  Created by Pablo Pérez on 13/10/12.
//  Copyright (c) 2012 __MyCompanyName__. All rights reserved.
//

#import "KPANoticiasDetalleViewController.h"

#define CELL_CONTENT_WIDTH 320.0f
#define CELL_CONTENT_MARGIN 10.0f

@interface KPANoticiasDetalleViewController ()

@end

@implementation KPANoticiasDetalleViewController
@synthesize ntitle=_ntitle;
@synthesize description=_description;
@synthesize link=_link;
@synthesize pubDate=_pubDate;

- (id)initWithStyle:(UITableViewStyle)style
{
    self = [super initWithStyle:style];
    if (self) {
        // Custom initialization
    }
    return self;
}

- (void)viewDidLoad
{
    [super viewDidLoad];
    self.tableView=[[UITableView alloc] initWithFrame:self.view.bounds
style:UITableViewStyleGrouped];
}

- (void)viewDidUnload
{
    [super viewDidUnload];
    // Release any retained subviews of the main view.
    // e.g. self.myOutlet = nil;
}

-
(BOOL)shouldAutorotateToInterfaceOrientation:(UIInterfaceOrientation)interfaceOrientatio
```

```
n
{
    return (interfaceOrientation == UIInterfaceOrientationPortrait);
}

#pragma mark - Table view data source

- (NSInteger)numberOfSectionsInTableView:(UITableView *)tableView
{
    // Return the number of sections.
    return 1;
}

- (NSInteger)tableView:(UITableView *)tableView numberOfRowsInSection:(NSInteger)section
{
    // Return the number of rows in the section.
    return 4;
}

- (UITableViewCell *)tableView:(UITableView *)tableView
cellForRowAtIndexPath:(NSIndexPath *)indexPath
{
    UITableViewCell *cell = [tableView dequeueReusableCellWithIdentifier:@"Cell"];
    if (cell == nil) {
        cell = [[UITableViewCell alloc] initWithStyle:UITableViewCellStyleDefault
reuseIdentifier:@"Cell"];
    }

    // Configure the cell...
    cell.textLabel.numberOfLines=0;
    cell.textLabel.lineBreakMode=UILineBreakModeWordWrap;

    switch (indexPath.row) {
        case 0:
            cell.textLabel.text=self.ntitle;
            cell.textLabel.font = [UIFont fontWithName:@"Verdana-Bold" size:12];
            cell.selectionStyle = UITableViewCellSelectionStyleNone; //No se podrá
pulsar sobre esta celda
            break;

        case 1:
            cell.textLabel.text=self.pubDate;
            cell.textLabel.font = [UIFont fontWithName:@"Verdana-Italic" size:11];
            cell.selectionStyle = UITableViewCellSelectionStyleNone; //No se podrá
pulsar sobre esta celda
            break;

        case 2:
            cell.textLabel.text=self.description;
            cell.textLabel.font = [UIFont fontWithName:@"Verdana" size:13];
            cell.selectionStyle = UITableViewCellSelectionStyleNone; //No se podrá
pulsar sobre esta celda
            break;

        case 3:
            cell.textLabel.text=@"Ver noticia en Safari";
            cell.textLabel.font = [UIFont fontWithName:@"Verdana-Bold" size:11];
            cell.textLabel.textAlignment=UITextAlignmentCenter;
            cell.textLabel.textColor=[UIColor purpleColor];
            break;
    }

    return cell;
}

// Función para que el alto de la celda se reajuste con el tamaño del contenido
-(CGFloat)tableView:(UITableView *)tableView heightForRowAtIndexPath:(NSIndexPath
*)indexPath{
    if (indexPath.row==2) {
        CGSize constraint = CGSizeMake(CELL_CONTENT_WIDTH - (CELL_CONTENT_MARGIN * 2),
20000.0f);
        CGSize size = [self.description sizeWithFont:[UIFont fontWithName:@"Verdana"
size:14] constrainedToSize:constraint lineBreakMode:UILineBreakModeWordWrap];
        CGFloat height = MAX(size.height, 44.0f);
        return height + (CELL_CONTENT_MARGIN * 2);
    }else if (indexPath.row==0) {
```

```
        CGSize constraint = CGSizeMake(CELL_CONTENT_WIDTH - (CELL_CONTENT_MARGIN * 2),
20000.0f);
        CGSize size = [self.ntitle sizeWithFont:[UIFont fontWithName:@"Verdana-Bold"
size:12] constrainedToSize:constraint lineBreakMode:UILineBreakModeWordWrap];
        CGFloat height = MAX(size.height, 44.0f);
        return height + (CELL_CONTENT_MARGIN * 2);
    }else {
        return 32.0f;
    }
}


#pragma mark - Table view delegate

- (void)tableView:(UITableView *)tableView didSelectRowAtIndexPath:(NSIndexPath
*)indexPath
{
    //Al pulsar sobre "Ver noticia en Safari" (row=3) se abrirá la página indicada en
self.link en Safari
    if (indexPath.row==3) {
        [[UIApplication sharedApplication]openURL:[NSURL URLWithString:self.link]];
    }
}

@end
```

```
//
//  KPAInfoViewController.h
//  UCProject
//
//  Created by Pablo Pérez on 24/09/12.
//  Copyright (c) 2012 __MyCompanyName__. All rights reserved.
//

#import <UIKit/UIKit.h>

@interface KPAInfoViewController : UIViewController
- (IBAction)sendEmail:(id)sender;
- (IBAction)openFB:(id)sender;
- (IBAction)openTwitter:(id)sender;
- (IBAction)openTuenti:(id)sender;
- (IBAction)openYT:(id)sender;

@end
```

```
//
//  KPAFichaViewController.h
//  UCProject
//
//  Created by Pablo Pérez on 23/09/12.
//  Copyright (c) 2012 __MyCompanyName__. All rights reserved.
//

#import <UIKit/UIKit.h>
#import <MapKit/MapKit.h>

@interface KPAFichaViewController : UIViewController<UITableViewDataSource,
UITableViewDelegate,UIAlertViewDelegate>{
    NSDictionary *_pageContents;
    NSString *_nombrePag;
    BOOL hasPerson;
}

@property (strong, nonatomic) NSDictionary *pageContents;
@property (strong, nonatomic) NSString *nombrePag;

@property (weak, nonatomic) IBOutlet UIScrollView *scrollView;
@property (weak, nonatomic) IBOutlet UILabel *subtitulo;
@property (weak, nonatomic) IBOutlet UILabel *persona;
@property (weak, nonatomic) IBOutlet UILabel *personaSubtitulo;
@property (weak, nonatomic) IBOutlet UIImageView *imagen;
@property (weak, nonatomic) IBOutlet UILabel *ubicacion;
@property (weak, nonatomic) IBOutlet MKMapView *mapa;
```

```
@property (weak, nonatomic) IBOutlet UITableView *tablaContacto;
- (IBAction)openInSafari:(id)sender;

@end
```

```
//
//  KPAFichaViewController.m
//  UCProject
//
//  Created by Pablo Pérez on 23/09/12.
//  Copyright (c) 2012 __MyCompanyName__. All rights reserved.
//

#import "KPAFichaViewController.h"
#import <QuartzCore/QuartzCore.h>

@interface KPAFichaViewController ()

@end

@implementation KPAFichaViewController
@synthesize pageContents=_pageContents;
@synthesize nombrePag=_nombrePag;
@synthesize scrollView = _scrollView;
@synthesize subtitulo = _subtitulo;
@synthesize persona = _persona;
@synthesize personaSubtitulo = _personaSubtitulo;
@synthesize imagen = _imagen;
@synthesize ubicacion = _ubicacion;
@synthesize mapa = _mapa;
@synthesize tablaContacto = _tablaContacto;

- (id)initWithNibName:(NSString *)nibNameOrNil bundle:(NSBundle *)nibBundleOrNil
{
    self = [super initWithNibName:nibNameOrNil bundle:nibBundleOrNil];
    if (self) {
        // Custom initialization
    }
    return self;
}

- (void)viewDidLoad
{
    [super viewDidLoad];
      // Do any additional setup after loading the view.
    [self loadContents:self.pageContents];
}

- (void)viewDidUnload
{
    [self setSubtitulo:nil];
    [self setPersona:nil];
    [self setPersonaSubtitulo:nil];
    [self setImagen:nil];
    [self setMapa:nil];
    [self setScrollView:nil];
    [self setPageContents:nil];
    [self setTablaContacto:nil];
    [self setUbicacion:nil];
    [super viewDidUnload];
    // Release any retained subviews of the main view.
}

-
(BOOL)shouldAutorotateToInterfaceOrientation:(UIInterfaceOrientation)interfaceOrientatio
n
{
    return (interfaceOrientation == UIInterfaceOrientationPortrait);
}

//// LOADING CONTENTS
-(void)checkSubtype:(NSString *)subtipo{
    if ([subtipo isEqualToString:@"VR"]) {
        hasPerson=YES;
    }else{
```

```
        hasPerson=NO;
    }
}

-(void)loadContents:(NSDictionary *)contents{
    [self checkSubtype:[contents objectForKey:@"subtipo"]];

    //Subtitulo
    if ([contents objectForKey:@"Nombre"]) {
        self.subtitulo.text=[contents objectForKey:@"Nombre"];
    }else{
        self.subtitulo.text=self.nombrePag;
    }

    //Imagen
    NSData * imageData = [[NSData alloc] initWithContentsOfURL: [NSURL
URLWithString:[contents objectForKey:@"Foto"]]];
    self.imagen.image = [UIImage imageWithData: imageData];
    [self.imagen.layer setBorderWidth:1.0];
    [self.imagen.layer setBorderColor:[[UIColor grayColor] CGColor]];

    //Persona
    if (hasPerson) {
        self.persona.text=[contents objectForKey:@"Persona"];
        [self setUILabelTextWithVerticalAlignTop:[contents objectForKey:@"Cargo"]];
        [self.imagen setFrame:CGRectMake(self.imagen.frame.origin.x,
self.imagen.frame.origin.y,120, self.imagen.frame.size.height)];
    }else{
        self.persona.text=@"";
        self.personaSubtitulo.text=@"";
        [self.imagen setFrame:CGRectMake(self.imagen.frame.origin.x,
self.imagen.frame.origin.y,300, self.imagen.frame.size.height)];
    }

    //Ubicacion - Los Centros no tienen campo "Ubica" --> Se añade al label el nombre de
la pagina
    if ([contents objectForKey:@"Ubica"]) {
        self.ubicacion.text=[contents objectForKey:@"Ubica"];
    }else if ([[contents objectForKey:@"subtipo"] isEqualToString:@"Centro"]){
        self.ubicacion.text=self.nombrePag;
    }else{
        self.ubicacion.text=@"";
    }


    //Mapa
    [self.mapa.layer setBorderWidth:1.0];
    [self.mapa.layer setBorderColor:[[UIColor grayColor] CGColor]];
    if ([contents objectForKey:@"Mapa"]) {
        NSString *coords= [contents objectForKey:@"Mapa"];
        NSRange range = [coords rangeOfString:@","];
        [self updateMap:[[coords substringToIndex:range.location] floatValue]
Longitude:[[coords substringFromIndex:range.location+1] floatValue]];
    }

    //ScrollView
    [self.scrollView setContentSize:CGSizeMake(320, 650)];

    //Table
    self.tablaContacto.backgroundColor=[UIColor clearColor];
    [self.tablaContacto.layer setBorderWidth:1.0];
    [self.tablaContacto.layer setBorderColor:[[UIColor grayColor] CGColor]];
    self.tablaContacto.layer.cornerRadius=8.0;
}

//LABEL
- (void)setUILabelTextWithVerticalAlignTop:(NSString *)theText {
    CGSize labelSize = CGSizeMake(168, 115);
    CGSize theStringSize = [theText sizeWithFont:self.personaSubtitulo.font
constrainedToSize:labelSize lineBreakMode:self.personaSubtitulo.lineBreakMode];
    self.personaSubtitulo.frame = CGRectMake(self.personaSubtitulo.frame.origin.x,
self.personaSubtitulo.frame.origin.y, theStringSize.width, theStringSize.height);
    self.personaSubtitulo.text = theText;
}

//TABLA
```

```
#pragma mark - Table view data source

- (NSInteger)numberOfSectionsInTableView:(UITableView *)tableView
{
    // Return the number of sections.
    return 1;
}

- (NSInteger)tableView:(UITableView *)tableView numberOfRowsInSection:(NSInteger)section
{
    // Return the number of rows in the section.
    return 3;
}


- (UITableViewCell *)tableView:(UITableView *)tableView
cellForRowAtIndexPath:(NSIndexPath *)indexPath
{
    UITableViewCell *cell = [tableView dequeueReusableCellWithIdentifier:@"Cell"];

    if (cell == nil) {
        cell = [[UITableViewCell alloc] initWithStyle:UITableViewCellStyleValue1
reuseIdentifier:@"Cell"];
    }

    // Configure cell
    cell.textLabel.font = [UIFont fontWithName:@"ArialMT" size:14];
    cell.detailTextLabel.font = [UIFont fontWithName:@"ArialMT" size:14];

    switch (indexPath.row) {
        case 0:
            cell.textLabel.text=@"Teléfono";
            if ([self.pageContents objectForKey:@"Telefono"]) {
                cell.detailTextLabel.text=[self.pageContents objectForKey:@"Telefono"];
            }else{
                cell.detailTextLabel.text=@"";
            }
            cell.imageView.image=[UIImage imageNamed:@"phone.png"];
            break;

        case 1:
            cell.textLabel.text=@"Fax";
            if ([self.pageContents objectForKey:@"Fax"]) {
                cell.detailTextLabel.text=[self.pageContents objectForKey:@"Fax"];
            }else{
                cell.detailTextLabel.text=@"";
            }
            cell.imageView.image=[UIImage imageNamed:@"fax.png"];
            // Esta celda no se podrá pulsar
            cell.selectionStyle = UITableViewCellSelectionStyleNone;
            break;

        case 2:
            cell.textLabel.text=@"Email";
            if ([self.pageContents objectForKey:@"Email"]) {
                cell.detailTextLabel.text=[self.pageContents objectForKey:@"Email"];
            }else{
                cell.detailTextLabel.text=@"";
            }
            cell.imageView.image=[UIImage imageNamed:@"mail.png"];
            break;
    }
    return cell;
}

- (UIView *)tableView:(UITableView *)tableView viewForHeaderInSection:(NSInteger)section
{
      // TITULO "Contacto" customizado de la tablaContacto
      UIView* customView = [[UIView alloc] initWithFrame:CGRectMake(0.0, 0.0, 300.0,
22.0)];
      UILabel * headerLabel = [[UILabel alloc] initWithFrame:CGRectZero];
      headerLabel.backgroundColor = [UIColor grayColor];
      headerLabel.opaque = YES;
      headerLabel.textColor = [UIColor whiteColor];
      headerLabel.highlightedTextColor = [UIColor darkGrayColor];
      headerLabel.font = [UIFont fontWithName:@"Arial-BoldMT" size:14];
```

```
      headerLabel.frame = CGRectMake(0.0, 0.0, 300.0, 21.0);
      headerLabel.text = @"  Contacto";// i.e. array element
      [customView addSubview:headerLabel];

      return customView;
}

- (void)tableView:(UITableView *)tableView didSelectRowAtIndexPath:(NSIndexPath
*)indexPath
{
    UITableViewCell *cell = [self.tablaContacto cellForRowAtIndexPath:indexPath];

    if ([cell.detailTextLabel.text isEqualToString:@""]==NO) {
        switch (indexPath.row) {
            case 0:
                //TELF
                [self showCallAlert:cell.detailTextLabel.text];
                break;

            case 2:
                //MAIL
                [[UIApplication sharedApplication]openURL:[NSURL URLWithString:[NSString
stringWithFormat:@"mailto:%@",cell.detailTextLabel.text]]];
                break;
        }
    }
}

// DRAW LOCATION ON MAP
- (void) updateMap:(float)lat Longitude:(float)lon{
    CLLocationCoordinate2D  point;
    point.latitude = lat;
    point.longitude =lon;

    MKCoordinateRegion region;
    MKCoordinateSpan span;
    span.latitudeDelta=0.002;
    span.longitudeDelta=0.002;
    region.span=span;
    region.center=point;

    MKPointAnnotation *marker = [[MKPointAnnotation alloc] init];
    [marker setCoordinate:point];
    [marker setTitle:self.nombrePag];
    [self.mapa addAnnotation:marker];
    [self.mapa setRegion:region animated:YES];
}

//Open in Safari
- (IBAction)openInSafari:(id)sender {
    [[UIApplication sharedApplication]openURL:[NSURL URLWithString:[self.pageContents
objectForKey:@"Enlace"]]];
}

-(void)callPhone{
    NSIndexPath *indexPath = [self.tablaContacto indexPathForSelectedRow];
    UITableViewCell *cell = [self.tablaContacto cellForRowAtIndexPath:indexPath];
    [[UIApplication sharedApplication]openURL:[NSURL URLWithString:[NSString
stringWithFormat:@"tel:%@",cell.detailTextLabel.text]]];
}

//// SHOW ALERT
-(void) showCallAlert:(NSString *)telefono{
    UIAlertView *alert = [[UIAlertView alloc]
                          initWithTitle:@"Contacto"
                          message:[NSString stringWithFormat:@"¿Desea llamar al %@
(%@)?",telefono,self.nombrePag]
                          delegate:self
                          cancelButtonTitle:@"Cancelar"
                          otherButtonTitles:@"Llamar", nil];
    [alert show];
}

-(void)alertView:(UIAlertView *)alertView clickedButtonAtIndex:(NSInteger)buttonIndex{
    if (buttonIndex == 1)
    {
```

```
        [self callPhone];
    }
}

@end
```

```
//
//  KPAMapaViewController.h
//  UCProject
//
//  Created by Pablo Pérez on 24/09/12.
//  Copyright (c) 2012 __MyCompanyName__. All rights reserved.
//

#import <UIKit/UIKit.h>
#import <MapKit/MapKit.h>

@interface KPAMapaViewController : UIViewController{
    NSArray *_mapaContents;
    NSString *_nombreMapa;
    NSMutableArray *_latitudes;
    NSMutableArray *_longitudes;
    float latCentro;
    float lonCentro;
    BOOL finderActivated;
}

@property (strong, nonatomic) NSArray *mapaContents;
@property (strong, nonatomic) NSString *nombreMapa;
@property (strong, nonatomic) NSMutableArray *latitudes;
@property (strong, nonatomic) NSMutableArray *longitudes;
@property (weak, nonatomic) IBOutlet UILabel *mapaTitulo;
@property (weak, nonatomic) IBOutlet MKMapView *mapa;
- (IBAction)setStandardView:(id)sender;
- (IBAction)setHybridView:(id)sender;
- (IBAction)findUser:(id)sender;

@end
```

```
//
//  KPAMapaViewController.m
//  UCProject
//
//  Created by Pablo Pérez on 24/09/12.
//  Copyright (c) 2012 __MyCompanyName__. All rights reserved.
//

#import "KPAMapaViewController.h"
#import <QuartzCore/QuartzCore.h>

#define MAP_PADDING 1.1
#define MINIMUM_VISIBLE_LATITUDE 0.01

@interface KPAMapaViewController ()

@end

@implementation KPAMapaViewController
@synthesize mapaContents=_mapaContents;
@synthesize nombreMapa=_nombreMapa;
@synthesize mapaTitulo = _mapaTitulo;
@synthesize mapa = _mapa;
@synthesize latitudes=_latitudes;
@synthesize longitudes=_longitudes;

- (id)initWithNibName:(NSString *)nibNameOrNil bundle:(NSBundle *)nibBundleOrNil
{
    self = [super initWithNibName:nibNameOrNil bundle:nibBundleOrNil];
    if (self) {
        // Custom initialization
    }
    return self;
```

```
}

- (void)viewDidLoad
{
    [super viewDidLoad];

      // Initial values
    finderActivated=NO;
    self.latitudes=[NSMutableArray new];
    self.longitudes=[NSMutableArray new];

    // Initial operations
    self.mapaTitulo.text=self.nombreMapa;
    [self.mapa.layer setBorderWidth:1.0];
    [self.mapa.layer setBorderColor:[[UIColor grayColor] CGColor]];
    [self centerMapOnMarkers:self.mapaContents];
    [self loadContents:self.mapaContents];
}

- (void)viewDidUnload
{
    [self setMapaTitulo:nil];
    [self setMapa:nil];
    [super viewDidUnload];
    // Release any retained subviews of the main view.
    self.mapaContents=nil;
    //Borrar marcadores del mapa
    [self.mapa removeAnnotations:self.mapa.annotations];
}

-
(BOOL)shouldAutorotateToInterfaceOrientation:(UIInterfaceOrientation)interfaceOrientatio
n
{
    return (interfaceOrientation == UIInterfaceOrientationPortrait);
}

// LOAD CONTENTS
-(void)loadContents:(NSArray *)contents{
    for(int i=0;i<contents.count;i++){
        NSDictionary *tmp=[contents objectAtIndex:i];
        [self setMarker:[[tmp objectForKey:@"Latitud"] floatValue] Longitude:[[tmp
objectForKey:@"Longitud"] floatValue] Texto:[tmp objectForKey:@"Nombre_Mapa"]];
    }
}

// DRAW LOCATION ON MAP
-(void)centerMapOnMarkers:(NSArray *)mapaArray{
    NSDictionary *tmp=[mapaArray objectAtIndex:0];
    float minLat=[[tmp objectForKey:@"Latitud"] floatValue];
    float maxLat=[[tmp objectForKey:@"Latitud"] floatValue];
    float minLon=[[tmp objectForKey:@"Longitud"] floatValue];
    float maxLon=[[tmp objectForKey:@"Longitud"] floatValue];

    for(int i=1;i<mapaArray.count;i++){
        tmp=[mapaArray objectAtIndex:i];
        if ([[tmp objectForKey:@"Latitud"] floatValue]<minLat) {
            minLat=[[tmp objectForKey:@"Latitud"] floatValue];
        }else if([[tmp objectForKey:@"Latitud"] floatValue]>maxLat){
            maxLat=[[tmp objectForKey:@"Latitud"] floatValue];
        }
        if ([[tmp objectForKey:@"Longitud"] floatValue]<minLon) {
            minLon=[[tmp objectForKey:@"Longitud"] floatValue];
        }else if([[tmp objectForKey:@"Longitud"] floatValue]>maxLon){
            maxLon=[[tmp objectForKey:@"Longitud"] floatValue];
        }
    }

    MKCoordinateRegion region;
    region.center.latitude = (minLat + maxLat) / 2;
    region.center.longitude = (minLon + maxLon) / 2;
    region.span.latitudeDelta = (maxLat - minLat) * MAP_PADDING;
    region.span.latitudeDelta = (region.span.latitudeDelta < MINIMUM_VISIBLE_LATITUDE)?
MINIMUM_VISIBLE_LATITUDE : region.span.latitudeDelta;
    region.span.longitudeDelta = (maxLon - minLon) * MAP_PADDING;
    MKCoordinateRegion scaledRegion = [self.mapa regionThatFits:region];
```

```
    [self.mapa setRegion:scaledRegion animated:YES];
}

- (void) setMarker:(float)lat Longitude:(float)lon Texto:(NSString *)texto{
    CLLocationCoordinate2D  point;
    point.latitude = lat;
    point.longitude =lon;

    MKPointAnnotation *marker = [[MKPointAnnotation alloc] init];
    [marker setCoordinate:point];
    [marker setTitle:texto];
    [self.mapa addAnnotation:marker];
}

- (IBAction)setStandardView:(id)sender {
    self.mapa.mapType=MKMapTypeStandard;
}

- (IBAction)setHybridView:(id)sender {
    self.mapa.mapType=MKMapTypeHybrid;
}

- (IBAction)findUser:(id)sender {
    NSDictionary *dic=[NSDictionary
dictionaryWithObjectsAndKeys:@"Usuario",@"Nombre_Mapa",[NSString
stringWithFormat:@"%f",self.mapa.userLocation.coordinate.latitude],@"Latitud",[NSString
stringWithFormat:@"%f",self.mapa.userLocation.coordinate.longitude],@"Longitud", nil];
    NSMutableArray *tmp=[NSMutableArray arrayWithArray:self.mapaContents];
    [tmp addObject:dic];
    [self centerMapOnMarkers:tmp];
}
@end
```

```
//
//  KPATextoHTMLViewController.h
//  UCProject
//
//  Created by Pablo Pérez on 10/10/12.
//  Copyright (c) 2012 __MyCompanyName__. All rights reserved.
//

#import <UIKit/UIKit.h>

@interface KPATextoHTMLViewController : UIViewController{
    NSMutableString *_htmlContent;
}
@property (weak, nonatomic) IBOutlet UIWebView *webView;
@property (strong, nonatomic) NSMutableString *htmlContent;

@end
```

```
//
//  KPATextoHTMLViewController.m
//  UCProject
//
//  Created by Pablo Pérez on 10/10/12.
//  Copyright (c) 2012 __MyCompanyName__. All rights reserved.
//

#import "KPATextoHTMLViewController.h"

@interface KPATextoHTMLViewController ()

@end

@implementation KPATextoHTMLViewController
@synthesize webView;
@synthesize htmlContent=_htmlContent;

- (id)initWithNibName:(NSString *)nibNameOrNil bundle:(NSBundle *)nibBundleOrNil
{
```

```
    self = [super initWithNibName:nibNameOrNil bundle:nibBundleOrNil];
    if (self) {
        // Custom initialization
    }
    return self;
}

- (void)viewDidLoad
{
    [super viewDidLoad];
      // Do any additional setup after loading the view.
    [self.webView loadHTMLString:self.htmlContent baseURL:nil];
}

- (void)viewDidUnload
{
    [self setWebView:nil];
    [super viewDidUnload];
    // Release any retained subviews of the main view.
}

-
(BOOL)shouldAutorotateToInterfaceOrientation:(UIInterfaceOrientation)interfaceOrientatio
n
{
    return (interfaceOrientation == UIInterfaceOrientationPortrait);
}

@end
```